Explainable Multi-Objective Reinforcement Learning with Pareto-Conditioned Policies

Joanikij Chulev

Department of Computer Science, Bioengineering, Robotics and System Engineering (DIBRIS)

University of Genova

Supervisor

Dr. Hendrik Baier

Centrum Wiskunde and Informatica (CWI)

Eindhoven University of Technology (TU/e)

Prof. Simona Sacone

University of Genova (UniGe)

Master's Degree in Computer Engineering

2026

Acknowledgements

This project was carried out at Centrum Wiskunde & Informatica (CWI) in Amsterdam, within the Intelligent and Autonomous Systems group. I would like to thank CWI for providing a stimulating research environment and for giving me the opportunity to work on a project at the intersection of explainable artificial intelligence, multi-objective reinforcement learning, and autonomous decision-making systems. The research atmosphere at CWI greatly contributed to the development of this thesis and to my growth as a researcher.

First and foremost, I would like to thank my supervisor, Dr. Hendrik Baier, for his belief in me, guidance, feedback, and support during this research project. His expertise in reinforcement learning and artificial intelligence was extremely valuable in shaping the direction of this work, refining the research questions, and improving both the technical and scientific quality of the thesis. He showed me what a true research atmosphere should be like.

I am also deeply grateful to Prof. Simona Sacone for her academic supervision, support, and coordination from the University of Genova.

Finally, I would like to thank my family and friends for their continuous support, patience, and encouragement throughout my studies.

Abstract

Multi-objective reinforcement learning (MORL) addresses decisions involving conflicting objectives. Pareto-Conditioned Networks (PCNs) represent multiple trade-offs in a single policy by conditioning actions on a desired-return command. Although this command expresses the agent's objectives, the learned state–command–action mapping remains opaque. This thesis investigates the command as an explanatory interface. It makes four contributions. First, it introduces a return-only PCN that removes explicit horizon conditioning while achieving performance comparable to return-and-horizon conditioning across the evaluated environments. Second, it develops command-space counterfactual explanations that identify minimal command changes required for the same policy, in the same state, to prefer a specified alternative action. The proposed CF-ZOO method combines Pareto-front-informed initialization with black-box zeroth-order optimization, without requiring access to model internals. Third, it introduces a goal-influence signal that separates command influence from policy confidence and is validated against behavioural ground truth across diverse environments. Fourth, it provides an interactive testbed for inspecting rollouts, generating and verifying counterfactual explanations, and supporting human evaluations. Together, these contributions enable local, contrastive, and objective-level explanations through desired-return commands.

Contents

List of Figures

List of Tables

List of Acronyms

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ART	Adversarial Robustness Toolbox
AUROC	Area Under the Receiver Operating Characteristic Curve
CF-ZOO	Counterfactual Zeroth-Order Optimization
CI	Confidence Interval
CPU	Central Processing Unit
C&W	Carlini–Wagner
D^3PO	Decomposed, Diversity-Driven Policy Optimization
DST	Deep Sea Treasure
EU	Expected Utility
GPU	Graphics Processing Unit
GT	Ground Truth
HTML	Hypertext Markup Language
HV	Hypervolume
JS	Jensen–Shannon Divergence
JSON	JavaScript Object Notation
KL	Kullback–Leibler Divergence
LIME	Local Interpretable Model-Agnostic Explanations
MC	Monte Carlo

MDP	Markov Decision Process
MI	Mutual Information
MOMDP	Multi-Objective Markov Decision Process
MORL	Multi-Objective Reinforcement Learning
ND	Non-Dominated Points
PCN	Pareto-Conditioned Network
R	Return-Only PCN Variant
RH	Return-Plus-Horizon PCN Variant
RL	Reinforcement Learning
RQ	Research Question
SD	Standard Deviation
SHAP	Shapley Additive Explanations
SPSA	Simultaneous Perturbation Stochastic Approximation
TV	Total Variation Distance
UI	User Interface
WHITE-CW	White-Box Carlini–Wagner Baseline
XAI	Explainable Artificial Intelligence
XMORL	Explainable Multi-Objective Reinforcement Learning
XRL	Explainable Reinforcement Learning
ZOO	Zeroth-Order Optimization

Chapter 1

Introduction

1.1 Motivation

Deep learning provides expressive models for high-dimensional data [1; 2], and reinforcement learning (RL) extends such models to sequential decisions whose value depends on future consequences [3]. Deep RL has achieved strong results in domains including Atari and Go [4; 5], but the resulting neural decision rules are difficult to inspect. That difficulty matters beyond benchmark performance: an agent deployed in mobility, robotics, resource management, or industrial control may take actions whose consequences are delayed, cumulative, and costly to reverse. A user therefore needs more than the action itself. They may need to know which objective trade-off supported it, which alternative was considered, and how much the requested outcome would have to change before the policy behaved differently.

This opacity motivates explainable artificial intelligence (XAI), which asks how model outputs can be made understandable [6; 7]. LIME, SHAP, and Integrated Gradients address static predictions [8; 9; 10]; RL is harder because an action is one step in a trajectory. Its significance depends on the current state,

the learned policy, accumulated and future rewards, termination conditions, and environment dynamics. An explanation that accounts only for the immediate output can therefore miss why the action is useful over time or whether the same preference would matter at another state [11].

The problem becomes sharper when several objectives must be balanced. Real decision systems rarely optimize a single quantity: performance can conflict with safety, energy use, cost, fairness, comfort, robustness, or environmental impact. Reducing all objectives to one scalar reward can conceal these tensions and fixes a weighting before deployment, even when different users or situations require different compromises. Multi-objective reinforcement learning (MORL) instead represents rewards as vectors and seeks policies for different Pareto trade-offs while retaining each objective's contribution to the return [12; 13]. The resulting objective structure is not merely an optimization detail. It supplies a vocabulary in which a contrast can be stated directly: an alternative action may require accepting less of one objective in exchange for more of another. Making that structure explicit can therefore support contrastive explanation [11; 14].

The interpretability literature stresses that an explanation must match its question and audience. Feature attribution links variables to one output, global summaries capture recurring policy patterns, and trajectory explanations connect decisions to longer-term outcomes [11; 14]. A developer may need a global account, whereas a user asking why one action was preferred to a foil needs a local contrast. In RL, changing the state or policy changes the subject of that explanation; a preference-based explanation should preserve both and intervene on an input meaningful to the user.

The multi-objective setting provides such an input but creates a further interpretability burden. Pareto fronts and coverage sets summarize policy trade-offs globally [12; 13], but do not show how a conditioned network maps one fixed

state and desired return to an action, whether a nearby command selects a foil, or whether the command matters there. Because objective-space proximity need not imply behavioural proximity under a nonlinear policy, explanation must bridge global objective geometry and the local decision surface. This bridge motivates the counterfactual, influence, and interaction components developed in the thesis.

Pareto-Conditioned Networks (PCNs) compactly represent many such trade-offs in one policy conditioned on a desired outcome [15]. At execution time, the desired-return command specifies which region of objective space the agent should pursue. Changing that command can change behaviour without retraining the network, which makes it a natural candidate for an interpretable control and explanation variable. It is already meaningful to a user in terms of desired outcomes rather than hidden activations or individual network weights.

However, conditioning on an interpretable input does not make the policy transparent [7; 11]. PCN conditions action selection on a remaining desired return [15], yet the local state–command–action mapping remains a neural black box. It exposes the requested trade-off without revealing whether the command influenced a decision or which nearby trade-off would produce a foil action. These gaps motivate the thesis: use the command as the interface, but supply methods that connect it to specific actions and measurable influence.

1.2 Research Problem

This thesis studies explanation methods that use the PCN command itself as the explanatory interface. The central object is the local mapping

$$(s_t, R_t) \;\mapsto\; a^\star, \tag{1.1}$$

where s_t is the current state, R_t the remaining desired-return command, and $a^\star$ the greedy action of the trained policy. Because the command encodes the desired multi-objective trade-off, interventions on the command produce explanations that are naturally expressed in the user's own preference language: *if your trade-off had shifted slightly towards objective X, the agent would have chosen Y.*

This requires a single interpretable intervention variable, a search suited to flat and non-convex black-box Command surfaces, a separate measure of whether the command influences a state at all, and software through which users can inspect and execute the resulting explanations.

1.3 Research Questions

The thesis addresses four research questions.

RQ1. Can the PCN policy be conditioned only on the state and the remaining desired return, so that the command becomes a single clean intervention variable for explanation, and how does such a return-only variant compare empirically with the original return-plus-horizon formulation when both are trained and evaluated under a shared protocol?

RQ2. How can local action decisions of a Pareto-conditioned policy be explained through minimal interventions on the desired-return command, and how reliably can a black-box search find such counterfactual commands on the learned command–action surface, compared with a white-box optimization baseline?

RQ3. How strongly does the desired-return command influence the policy's action distribution at each visited state, can this influence be separated using

insight from policy confidence, and can a query-based influence signal be validated against behavioural ground truth?

RQ4. How can command-space explanations be made inspectable and executable for a human user, including a protocol for realizing counterfactual commands in actual environment rollouts?

1.4 Contributions

The thesis makes four connected contributions, one for each research question.

1. **Return-only PCN design (Chapter 4).** The deployed policy is conditioned on state and remaining desired return, $\pi_\theta(a_t \mid s_t, R_t^{\text{des}})$. Reward-neutral suffix trimming and a preference for shorter near-equal trajectories retain temporal bias without a horizon input. A controlled comparison with the return-plus-horizon (RH) formulation across 21 environment configurations establishes broad comparability, not dominance.

2. **Command-space counterfactual explanations (Chapters 5 and 6).** Given a fixed state, command, selected action, and foil, the method seeks a minimal command change that makes the trained PCN prefer the foil. The black-box CF-ZOO search combines Pareto-front-seeded rays, binary boundary refinement, and ZOO-ADAM, and is evaluated against a white-box Carlini–Wagner baseline, a combinatorial stress test, and a dense-grid feasibility diagnosis.

3. **Goal-influence analysis (Chapter 7).** A per-state divergence between the command-conditioned distribution and a broad-generic same-state baseline separates influence from confidence. Total-variation, Jensen–Shannon, probe-flip, and reachable-centroid companions support validation against

command-grid and seeded Monte Carlo behavioural ground truth in ten environments.

4. **Interactive testbed (Chapter 8).** An application supports model and Pareto-front selection, live and replayable rollouts, foil queries with both explanation methods, and verified realization of counterfactual commands. It provides infrastructure for qualitative inspection and future human evaluation.

Parts of this work have been accepted at the Workshop on Explainable Artificial Intelligence (XAI) at IJCAI 2026 as the paper Command-Space Counterfactual Explanations for Pareto-Conditioned Reinforcement Learning (Chulev and Baier). Another similar version was also submitted to the AAAI-27 main conference.

The papers cover the counterfactual formulation and its evaluation; this thesis extends it with the return-only versus return-plus-horizon comparison, the goal-influence analysis, the interactive testbed, and thesis-level methodological detail and discussion.

1.5 Thesis Organization

Chapter 2 reviews the necessary foundations and research gap. Chapter 3 introduces the complete environment suite and identifies the subset used by each experiment. Chapter 4 compares return-only and return-horizon PCNs. Chapters 5 and 6 define and evaluate command-space counterfactuals; Chapter 7 develops the goal-influence analysis; and Chapter 8 presents the interactive system. Chapters 9 and 10 synthesize the findings, limitations, and future work. The appendices provide hyperparameters and extended results.

Chapter 2

Background and Related Work

This chapter introduces the required technical background and positions the thesis within related work.

2.1 Reinforcement Learning

Reinforcement learning (RL) models sequential interaction: an agent observes a state, selects an action, receives a reward, and seeks a policy maximizing expected long-term return [3]. The standard Markov decision process (MDP) is $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, with state and action spaces, transition function $P(s' \mid s, a)$, scalar reward $r(s, a, s')$, and discount factor $\gamma \in [0, 1]$. Under the Markov assumption, the current state suffices to model the next-state distribution given the action. Neural policies and value functions enable high-dimensional deep RL [4; 5], but obscure the resulting decision rules.

2.2 Multi-Objective Reinforcement Learning

Multi-objective reinforcement learning (MORL) replaces a scalar reward with a vector when decisions must balance several objectives. A multi-objective Markov decision process (MOMDP) is

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mathbf{r}, \gamma)\,, \tag{2.1}$$

where $\mathbf{r}(s, a, s') \in \mathbb{R}^m$ is an m-dimensional reward vector and the remaining elements are as in the scalar MDP [12; 13; 16]. For a trajectory τ, the return is vector-valued:

$$\mathbf{G}(\tau) = \sum_{t=0}^{T} \gamma^t \, \mathbf{r}(s_t, a_t, s_{t+1}). \tag{2.2}$$

Pareto Dominance and Coverage Sets. A return vector $\mathbf{x}$ *Pareto-dominates* $\mathbf{y}$ when it is at least as good in every objective and strictly better in at least one. The *Pareto front* contains the attainable returns that no other attainable return dominates. If utility is unknown during training, a *coverage set* represents policies that are optimal for different utilities [12]. Fixed scalarization, by contrast, chooses one preference in advance. A linear scalarization can miss non-convex regions of the Pareto front [12; 13]. It can also hide the objective structure and leave preference specification to the designer [13].

Solution-Set Metrics. Chapter 4 compares learned solution sets with two standard quantities.

Hypervolume. For achieved returns $X \subset \mathbb{R}^m$ and a dominated reference point $\mathbf{z}_{\text{ref}}$, hypervolume is the measure of the region dominated by X and bounded by that point [13]. Larger values indicate greater convergence and spread; comparisons require a fixed reference point.

Expected utility. This is the expectation, over a utility distribution, of the best utility attainable in the solution set [12; 13]. Here utilities are linear and weights follow the protocol in Section 4.4.

Deep and Applied MORL. Deep MORL uses neural approximation in complex environments [17; 18]; related work approximates continuous Pareto frontiers with policy gradients [19] and connects multi-objective and multi-task RL through generalized policy improvement and preference adaptation [20]. Because MORL exposes rather than hides trade-offs, explanations can ask how those trade-offs influence behaviour.

2.3 Pareto-Conditioned and Preference-Conditioned Policies

Preference- and return-conditioned methods avoid storing a policy for every trade-off by replacing $\pi_\theta(a \mid s)$ with one policy $\pi_\theta(a \mid s, c)$ conditioned on a preference, weight, or desired return c.

The Original PCN Formulation. The original Pareto-Conditioned Network (PCN) conditions one policy on the state, desired return, and desired horizon, so one network can represent multiple requested outcomes and time budgets [15]:

$$\pi_\theta\big(a_t \mid s_t, R_t^{\text{des}}, h_t^{\text{des}}\big)\,, \tag{2.3}$$

where R_t^{des} is the requested multi-objective return and h_t^{des} the requested remaining steps. PCN stores achieved returns, horizons, transitions, and actions in a bounded replay buffer biased towards non-dominated outcomes, then learns by conditional behavioural cloning [15]. The original work does not isolate the hori-

zon's role; Chapter 4 tests a return-only alternative. Pareto conditioning has also been extended to continuous actions [21].

Related Conditioned-Policy Methods. Related methods condition agents on changing scalarization weights [22], learn universal preference-conditioned networks [23], or use hypernetworks for dense Pareto-set coverage [24]. Preference-Controllable RL targets controllability [25]; D3PO addresses gradient interference and representational collapse [26]; and envelope Q-learning generalizes one network over linear utilities [27].

These methods make the command a semantic runtime interface. A local explanation can therefore identify the smallest command change that changes an action.

2.4 Explainable Reinforcement Learning

Explainable reinforcement learning (XRL) includes intrinsic and post-hoc, local and global explanations of states, actions, trajectories, rewards, policies, and models [11; 28; 29; 30]. Unlike one-step classification, RL agents act repeatedly and affect later states. An explanation may concern an action, its influential states or rewards, a trajectory, or the policy's general strategy. These targets answer different questions and should not be treated as interchangeable [11; 31].

A local explanation concerns one decision in a specified context, whereas a global explanation summarizes behaviour across many states. Intrinsic methods make the policy itself interpretable; post-hoc methods explain a trained policy without changing its decision rule. Explanations may additionally describe associations in the policy, contrast alternatives, or connect behaviour to an agent's goals and dispositions [14; 30]. The appropriate form consequently depends on whether the user wants to understand an immediate choice, a longer-term out-

come, or the overall policy.

Within XRL, this thesis addresses the gap of explaining fixed-state actions through minimal changes to the objective command.

2.5 Counterfactual Explanations

Counterfactuals in Supervised Learning. A counterfactual states how an input must change to produce a different output [32]; it is local and contrastive by construction. Supervised methods optimize validity and proximity, sometimes adding sparsity, manifold closeness [33], diversity [34], actionability, or causal consistency [35; 36]. Prototype guidance can further encourage manifold proximity by steering a counterfactual towards a representative prototype of the target class [37].

Why Sequential Settings Are Different. RL adds temporality, dynamics, and stochasticity: a nearby state may be unreachable, and recourse may fail across rollouts [38]. RL-specific desiderata therefore include reachability, stochastic recourse, and actionability under environment constraints; Chapter 5 specializes all seven surveyed desiderata to command space.

Counterfactual Methods in RL. RL counterfactuals modify visual states through generative models [39], propose alternative action sequences or policies under uncertainty [40], or search reachable trajectories for states where a foil is likely [41]. ACTER produces diverse actionable action sequences [42]; COViz compares outcomes visually [43]; COUNTERPOL constructs a minimally changed policy whose behaviour provides the counterfactual contrast [44]; and continuous-action work changes action sequences [45]. GANterfactual-RL and SAFE-RL produce plausible visual counterfactuals using generative or saliency

cues [46; 47]; CausalCF applies causal counterfactuals to robotic robustness [48].

In contrast, this thesis holds pixels, state variables, action history, and policy parameters fixed, and changes the requested objective trade-off to induce a local foil action.

2.6 Explainable Multi-Objective Reinforcement Learning

Explainable MORL (XMORL) builds on reward decomposition for action preferences [49], trade-off rationales in multi-objective planning [50], and contrastive explanations that teach tactical rules [51]. XMORL surveys identify preference adaptation and competing goals as challenges beyond scalar-reward XRL [52; 53].

Recent methods cluster policy sets in objective and behavioural spaces [54], structure parameter and performance spaces for interpretability [55], and connect preference-conditioned trajectory segments to Pareto trade-offs [56]. Diagnostics also measure global preference–return controllability [57] or compare front geometry with trajectory representations [58]. These policy-set, parameter, trajectory, and episode-level approaches do not provide a fixed-state command counterfactual for a specified foil action.

2.7 Adversarial and Zeroth-Order Optimization

Counterfactual search and adversarial perturbation both seek small input changes that alter model output. Carlini–Wagner attacks balance proximity against a targeted loss [59], while DeepFool approximates decision boundaries iteratively [60]. Black-box ZOO estimates coordinate-wise gradients with finite differences [61]; SPSA estimates a multivariate gradient from two simultaneous perturbation eval-

uations [62]; and Natural Evolution Strategies update a search distribution toward higher-fitness regions [63]. Cheng et al. instead optimize directions and scalar boundary distances in hard-label settings [64], inspiring Chapter 5's boundary-seeded stage.

RL policies are vulnerable to perturbed observations [65] and adversarial policies acting through natural interactions [66]. Attacks also target trading [67], recommender systems [68], lane changing [69], and selected key decisions [70]. This thesis repurposes their optimization logic for explanation: it perturbs the PCN command to expose a local action boundary, not to attack the agent.

2.8 Research Gap

The reviewed work covers RL counterfactuals over states, observations, action sequences, and policies; conditioned MORL policies and policy sets; and XMORL explanations at reward, trade-off, policy-set, trajectory, and global-control levels. It does not directly answer this fixed-state question:

> Given a trained Pareto-conditioned policy, a fixed state, an original desired-return command, and a foil action, what minimal change in the command would cause the policy to select the foil action?

Answering it requires a single command intervention (Chapter 4), black-box search over the learned command–action surface (Chapters 5 and 6), a complementary influence measure (Chapter 7), and an interface for inspection and execution (Chapter 8), including a human interaction interface.

Chapter 3

Experimental Environments

This chapter introduces the complete environment suite before it is used in the experimental chapters. The suite contains 21 configured tasks drawn from 14 environment families: four diagnostics created for this thesis, eight dimensional variants of Walkroom, and nine published MORL or control benchmarks. Every environment returns a vector reward and every coordinate is maximized; quantities described as costs or penalties therefore enter the return with a negative sign. Published environments are instantiated through the MO-Gymnasium and Gymnasium interfaces [71; 72]. The exact versions and modifications used here are important: Deep Sea Treasure uses the concave map, Fruit-Tree has depth six, Minecart is deterministic, Mountain Car uses the time–speed reward, and the control tasks are MO-Lunar-Lander-v3 and MO-Reacher-v5.

The 21 configurations serve different experimental roles. The return-only versus return-plus-horizon comparison in Chapter 4 uses all of them. The counterfactual experiments in Chapter 6 use Branch-Path, Breakable-Bottles, Collect-Two, Deep Sea Treasure, Fruit-Tree, Minecart, Resource-Gathering, and Reward-Line. The goal-influence analysis in Chapter 7 uses those eight environments and adds Four-Room and Three-Tree, giving ten environments in total. The interactive

system in Chapter 8 bundles both policy variants for the 13 non-Walkroom environments and for Walkroom-2 and Walkroom-3, giving 15 selectable environment configurations.

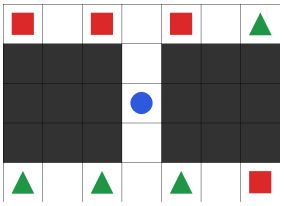
Custom diagnostic environments. Branch-Path is a 5×7 grid whose central junction leads to two narrow branches containing A- and B-items. Collecting an item adds one unit to its corresponding objective; the 13-step limit makes the three trade-offs $(3, 1)$, $(2, 2)$, and $(1, 3)$ the attainable non-dominated outcomes. Collect-Two is a 7×7 grid with four surrounding landmarks and four reward coordinates. An episode ends after two distinct landmarks are collected: the first selected landmark yields 1.0 in its coordinate and the second yields 0.8, so the order matters and produces 12 non-dominated return vectors. Reward-Line is a 5×11 grid in which reaching column c of the terminal row yields $(1-u, u)$, where $u = c/10$; its 11 terminal cells expose a linear two-objective front. Three-Tree is a depth-four ternary decision tree. At each node the agent chooses one of the P, S, and E branches and receives a path-dependent three-objective reward; its $3^4 = 81$ terminal paths contain 14 non-dominated return vectors. These four environments were designed for this work, so their definitions and reward tables are the primary specification rather than adaptations of an external benchmark.

Walkroom. Walkroom is the scalable synthetic environment introduced with PCN [15]. In Walkroom-d, the state is a point in a d-dimensional lattice, the $2d$ actions move one step forward or backward along one coordinate, and either direction incurs -1 in that coordinate's reward. The episode ends when the final coordinate reaches a procedurally generated boundary determined by the other $d-1$ coordinates, so terminal locations trade coordinate-wise path lengths against one another. This thesis evaluates $d \in \{2, 3, 4, 5, 6, 7, 8, 9\}$, using side length 20 for $d \leq 5$, side length 10 for $d \geq 6$, and a 200-step limit.

Published discrete benchmarks. Concave Deep Sea Treasure is an 11×11

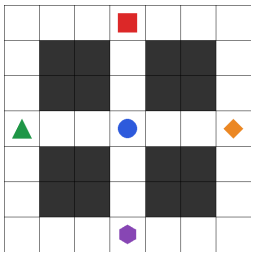
submarine grid with four movement actions; reaching a treasure terminates the episode, and the two objectives trade treasure value against a -1 time penalty per step [73]. Breakable-Bottles is a five-cell delivery problem with three actions and objectives for elapsed time, delivered bottles, and the potential side effect created by leaving a bottle in the path [74]. Resource-Gathering is a grid navigation task in which the agent can collect gold and a gem before returning home, while enemy cells introduce a risk of losing the collected resources; its objectives are the enemy penalty, gold, and gem [75]. Depth-6 Fruit-Tree is a full binary tree whose 64 leaves carry six-dimensional nutrient vectors for protein, carbohydrates, fats, vitamins, minerals, and water [27]. Four-Room is a 13×13 four-action grid in which the state records position and collected items; the three reward coordinates count blue squares, green triangles, and red circles before the agent reaches the goal [76; 77].

Published continuous-state benchmarks. Deterministic Minecart has a continuous kinematic state and six discrete controls for steering, acceleration, braking, mining, and idling; the agent trades the quantities of two ores against negative fuel consumption before returning to base [22]. MO-MountainCar-TimeSpeed retains the standard two-dimensional position–velocity state and three discrete controls, but decomposes reward into a -1 time objective and a positive speed objective [71; 72]. MO-Lunar-Lander-v3 has an eight-dimensional physical state and four discrete engine actions; its four objectives separate landing or crashing, flight shaping, main-engine fuel, and side-engine fuel [71; 72; 78]. MO-Reacher-v5 controls a two-joint arm from a six-dimensional angle-and-velocity observation. Its nine actions are the Cartesian combinations of negative, zero, or positive torque at each joint, and each of four rewards measures proximity of the fingertip to a different fixed target [71; 72; 77].

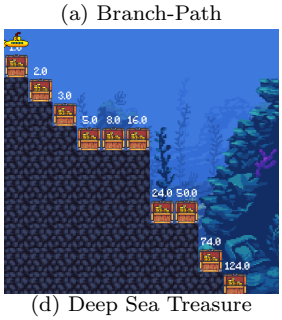


(a) Branch-Path

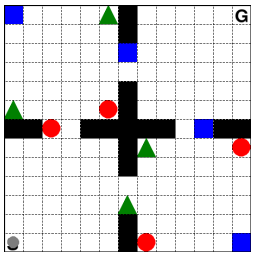
(b) Breakable-Bottles



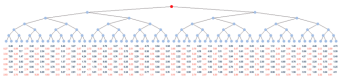
(c) Collect-Two



(d) Deep Sea Treasure

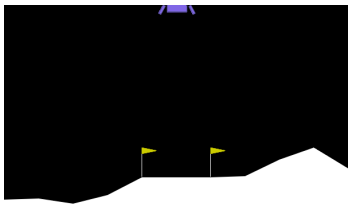


(e) Four-Room

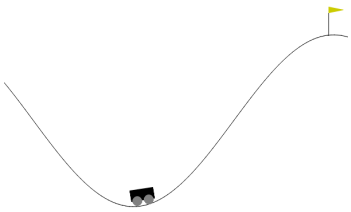


(f) Fruit-Tree

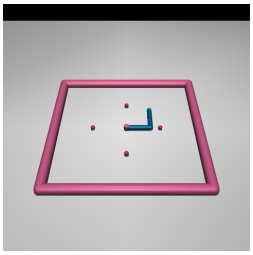
(g) Minecart



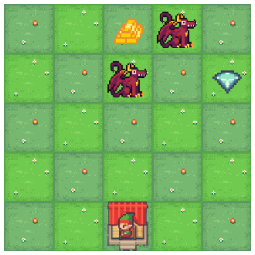
(h) Lunar-Lander



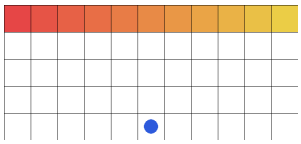
(i) Mountain-Car



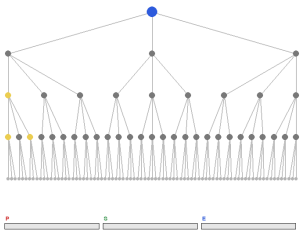
(j) MO-Reacher



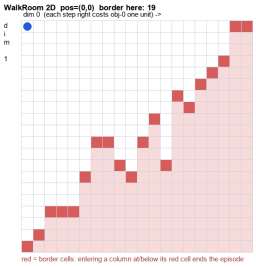
(k) Resource-Gathering



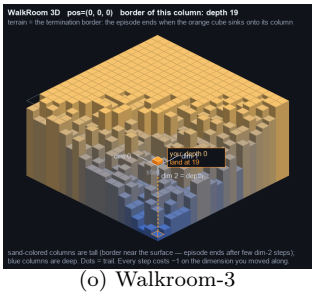
(l) Reward-Line



(m) Three-Tree



(n) Walkroom-2



(o) Walkroom-3

Figure 3.1: Start-state thumbnails for the 15 environments available in the interactive rollout system. Walkroom-2 and Walkroom-3 visualize the family used for all dimensional variants. Policies receive state vectors, not pixels.

Chapter 4

Return-Only Pareto-Conditioned Networks

This chapter defines a return-only PCN, explains how it retains temporal bias without a horizon input, and compares it with the original formulation across the 21 environment configurations introduced in Chapter 3.

4.1 The Original Return-Plus-Horizon Formulation

As reviewed in Section 2.3, the original PCN conditions one policy on state, desired return, and desired horizon [15]:

$$\pi_\theta\big(a_t \mid s_t, R_t^{\text{des}}, h_t^{\text{des}}\big)\,, \tag{4.1}$$

Training alternates data collection and supervised updates. A bounded replay buffer retains well-spread, non-dominated episodes; at step t, stored actions are relabelled with the return obtained from t onward and the remaining episode

length. New commands are sampled near current non-dominated returns. This return-plus-horizon variant is denoted RH.

The horizon disambiguates trajectories that achieve the same return at different speeds and provides an additional deployment control, but its role was not isolated experimentally in the original paper [15].

4.2 Why Isolate Return Conditioning

This thesis uses a revised return-only variant, abbreviated R, in which the deployed policy is

$$\pi_\theta\big(a_t \mid s_t, R_t^{\text{des}}\big)\,. \tag{4.2}$$

Two considerations motivate this choice without implying that horizon conditioning is universally useless or inferior.

Explanatory identifiability. The explanation objective of this thesis is to understand how desired-return commands affect action choice. Keeping the horizon would require counterfactuals over both return and time,

$$(R, h) \;\mapsto\; (R + \delta_R,\; h + \delta_h), \tag{4.3}$$

whereas the return-only formulation yields the simpler and semantically cleaner explanation problem

$$R \;\mapsto\; R + \delta_R. \tag{4.4}$$

Now every counterfactual component is expressed in objective units, isolating return-command effects from simultaneous temporal intervention.

Observed ambiguity of the horizon signal. In trained RH agents, changing only a desired horizon sometimes produced unstable or unintuitive actions even for Pareto-front return commands. This environment-specific observation supported removing horizon from the explanatory interface; it is not a general claim about RH.

4.3 Retained Temporal Inductive Biases

Without horizon, waiting or wandering can share return labels with efficient behaviour. Two training modifications retain useful temporal bias indirectly.

Reward-neutral suffix trimming. Before an episode is relabelled and stored, its reward-neutral suffix is removed. Formally, let $(r_0, \ldots, r_{T-1})$ be the reward vectors of an episode and define the suffix sums $\Sigma_k = \sum_{t=k}^{T-1} r_t$. The trimmed episode keeps steps $0, \ldots, k^\star - 1$, where $k^\star$ is the smallest index with $\Sigma_{k^\star} = \mathbf{0}$ (component-wise, within an absolute tolerance of 10^{-6}); if no such proper suffix exists, the episode is kept unchanged. Trimming prevents final stretches in which the agent no longer receives any additional reward, such as unnecessary waiting or wandering after the last collectible, from being treated as part of the intended command-conditioned behaviour.

Shorter-trajectory preference among near-equal returns. When commands are selected from the buffer for relabelling and for sampling new exploratory commands, stored returns are first filtered for Pareto non-dominance and then subjected to a time tie-break: if two stored episodes achieved near-equal return vectors, the longer one is removed from the command pool. Near-equality of returns $\mathbf{x}$ and $\mathbf{y}$ is tested component-wise as $|x_i - y_i| \leq \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}} \max(|x_i|, |y_i|, 1)$ with $\varepsilon_{\text{abs}} = 10^{-6}$ and $\varepsilon_{\text{rel}} = 10^{-3}$. This biases the replay-derived commands to-

wards the most efficient demonstration of each achieved outcome, which is precisely the disambiguation the horizon command provided in the RH formulation, without adding a second input dimension to the deployed policy.

4.4 Comparison and Conclusions

Shared protocol. One comparison script gives R and RH identical environment builders, step and update budgets, batch size, exploratory and initial episode counts, discount factor, replay capacity, and seed. They differ only in their command inputs (Eqs. (4.2) and (4.1)) and R's two inductive biases.

Deterministic evaluation greedily rolls out every final logged non-dominated command once. Four quantities summarize each variant:

Front. The number of stored, evaluated commands, i.e. the size of the final logged non-dominated command set that was rolled out.

ND. The number of Pareto non-dominated vectors among the *achieved* returns of those rollouts.

HV. Hypervolume measures how much objective space lies between the achieved non-dominated returns and a fixed poor reference point. R and RH use the same reference point within each environment, so the larger value indicates the better or broader achieved front. A return that is worse than the reference point in any objective lies outside this fixed comparison region and contributes no HV; it is excluded from the HV calculation but remains counted in Front and ND and remains available to EU. The implementation negates returns only because the software routine assumes minimization rather than maximization.

EU. Expected utility measures how useful the achieved returns are when the user's priorities are unknown. A weight vector represents one possible preference by assigning a non-negative importance to each objective, with all weights summing to one. The evaluation samples 10,000 such preferences uniformly using random seed 0. For each preference, it selects the achieved return with the highest weighted score. EU is the average of these 10,000 best scores. Within the same environment, a larger EU is better.

Thus Front measures retained commands, ND their non-dominated realized outcomes, HV front convergence and spread, and EU usefulness under unknown linear preferences.

Results across 21 environment configurations. Table 4.1 covers 21 custom and standard environments with 2 to 9 objectives. Under the evaluation's comparison rule, with no additional significance threshold, there are **13 ties, 5 R wins, and 3 RH wins**. Each environment was assessed independently using the full-precision results. Front is descriptive, whereas ND, HV, and EU measure the quality of the achieved set, with larger values preferred. A variant was assigned a win when these measures gave it an overall advantage, even slight; identical or extraordinarily similar results were recorded as ties.

		Return-only (R)				Return-plus-horizon (RH)			
Environment	Obj.	Front	ND	HV	EU	Front	ND	HV	EU
branch-path	2	3	3	6	2.495	3	3	6	2.495
breakable-bottles-v0	3	1	1	5015	13.15	1	1	5015	13.15
collect_two	4	12	12	0	0.738	12	12	0	0.738
dst	2	10	10	22,860	53.4	10	10	22,860	53.4
four-room-v0	3	1	1	8	2.0	2	2	5	1.75
fruit-tree-v0	6	62	62	12,280	5.058	61	61	12,020	5.058
minecart	3	22	22	198.8	0.458	22	18	196.4	0.458
mo-lunar-lander-v3	4	29	6	1.328×10^{11}	61.6	22	7	1.391×10^{11}	64.9
mo-mountaincar-timespeed	2	134	31	7,199	−15.96	118	27	7,238	−15.39
mo-reacher-v5	4	2500	2216	4.566×10^{7}	25.4	2500	2405	3.264×10^{7}	23.0
resource-gathering-v0	3	1	1	1.331	0.666	1	1	1.331	0.666
reward-line	2	11	11	0.45	0.748	11	8	0.43	0.723
three-tree	3	14	14	26.98	2.995	14	14	26.98	2.995
walkroom2	2	11	11	212	−4.57	11	11	212	−4.57
walkroom3	3	65	65	6,853	−2.13	65	65	6,857	−2.04
walkroom4	4	257	257	154,400	−1.11	252	252	154,300	−1.13
walkroom5	5	785	785	3.174×10^{6}	−0.732	857	857	3.176×10^{6}	−0.715
walkroom6	6	495	495	997,200	−0.244	535	531	997,359	−0.243
walkroom7	7	1080	1076	9.994×10^{6}	−0.185	1129	1124	9.994×10^{6}	−0.179
walkroom8	8	1841	1819	9.999×10^{7}	−0.147	1917	1888	9.999×10^{7}	−0.136
walkroom9	9	500	500	9.994×10^{8}	−0.185	500	500	9.995×10^{8}	−0.169

Table 4.1: Deterministic comparison of the return-only (R) and return-plus-horizon (RH) PCNs across 21 environment configurations under identical training and evaluation budgets. For each variant, every command in the final logged non-dominated set is rolled out greedily once. Obj. is the number of objectives; Front is the number of logged commands evaluated; and ND is the number of their achieved return vectors that remain Pareto non-dominated. HV measures the achieved non-dominated set within a fixed environment-specific reference region, while EU averages the best attainable linear utility over 10,000 sampled preference vectors. Larger HV and EU values are better, but they are comparable only between R and RH within the same environment because reward scales and HV reference points differ across environments. R and RH use the same reference point within each environment; achieved returns outside its region are excluded only from HV. EU is negative when the environment expresses outcomes as negative costs or penalties; it is not inherently bad, and a higher (less negative) EU still indicates better performance within that environment. The Front sizes of 500 for Walkroom9 and 2,500 for MO-Reacher reflect the bounded replay subset, not the true Pareto-front size or the achieved ND count.

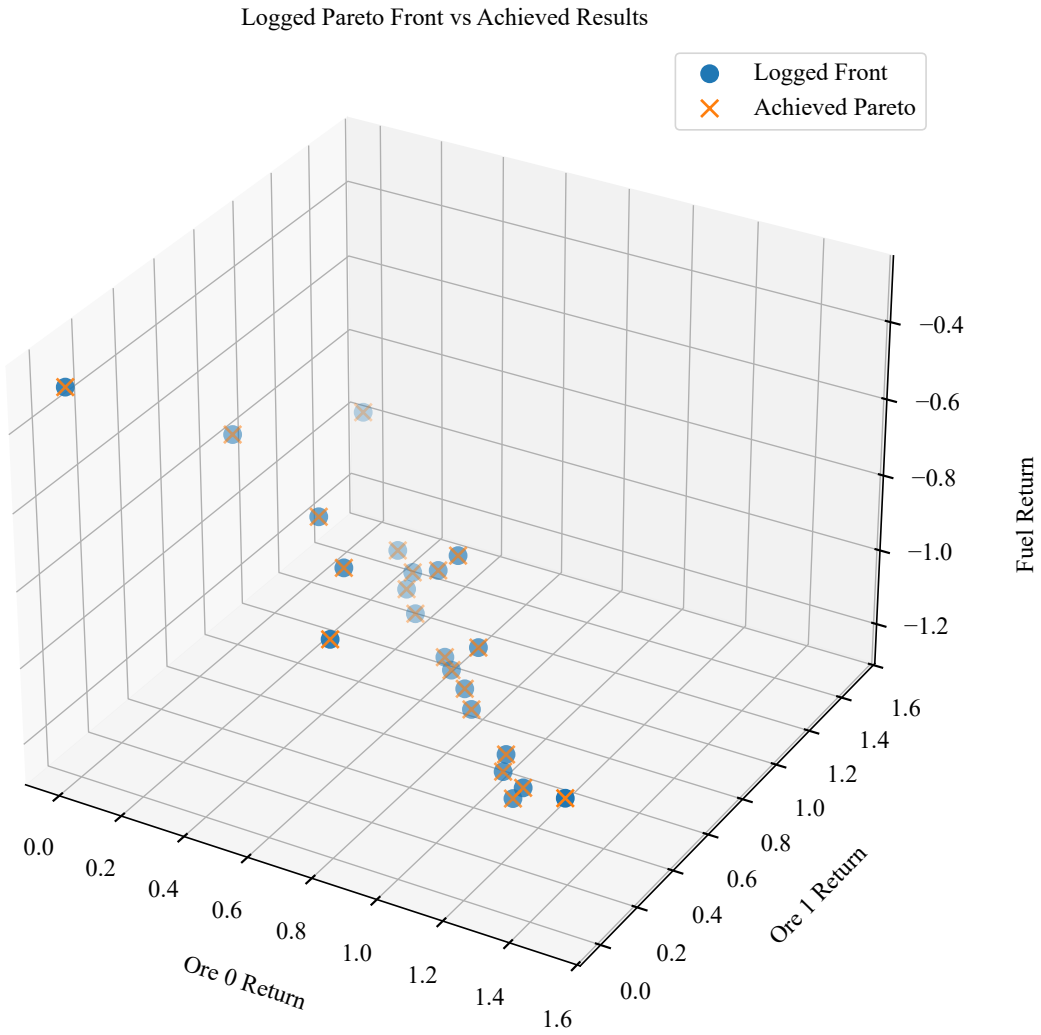


Figure 4.1: Pareto front recovered by the return-only PCN variant on Minecart. Blue points show the final logged learning front (the evaluation commands); orange crosses show the returns achieved when those commands are rolled out greedily. Axes are the two ore-return objectives and the fuel-return objective.

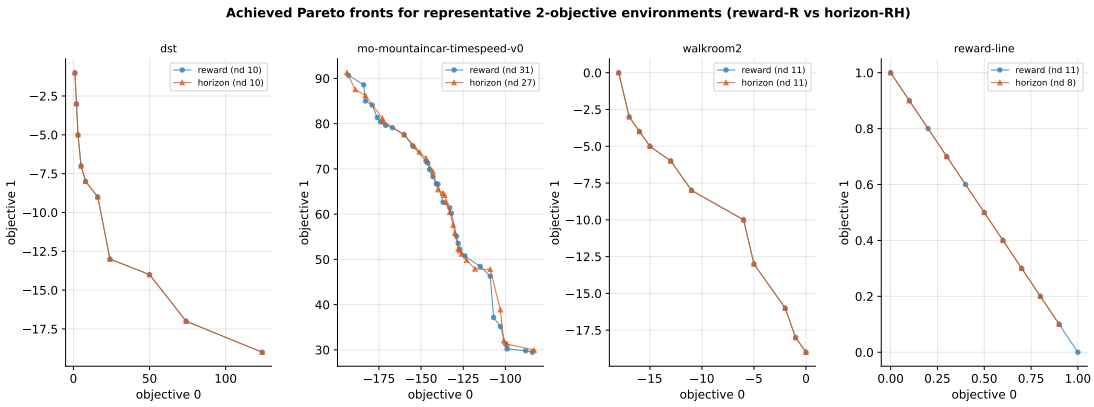


Figure 4.2: Achieved non-dominated return fronts of the return-only (R; blue circles) and return-plus-horizon (RH; orange triangles) PCNs in four representative two-objective environments. Each point is an achieved return from a greedy rollout of a final logged command, and each legend reports the number of achieved returns that remain non-dominated. The fronts nearly coincide on DST and Walkroom2, whereas R retains more non-dominated achieved points on MO-MountainCar (31 versus 27) and Reward-Line (11 versus 8).

Most environments, including all custom grids, show identical or near-identical metrics. R's wins largely reflect more achieved returns remaining non-dominated: Minecart has 22 versus 18 and Reward-Line 11 versus 8. Figure 4.1 shows that Minecart's achieved returns track its logged front, while Figure 4.2 directly compares the achieved R and RH fronts in four representative two-objective environments. Differences in algorithms, budgets, metrics, and reporting prevent comparison with prior PCN and Minecart results [15; 22]. RH wins concentrate in MO-Reacher and the 5–8-objective Walkrooms, where it retains larger fronts or slightly better HV or EU.

Conclusions. Under the shared protocol, R is broadly comparable to RH rather than superior. RH's high-dimensional advantages indicate that temporal conditioning can still help, but removing it with the two retained biases causes no systematic loss in these environments. This is sufficient for Chapters 5 and 7.

Chapter 5

Command-Space Counterfactuals

This chapter defines CF-ZOO, a black-box search for a bounded, minimal command perturbation that makes a foil action greedy at a fixed state. It combines a targeted margin loss, boundary-seeded directional initialization, and finite-difference refinement; a white-box Carlini–Wagner method provides a stronger-access baseline.

5.1 Counterfactual Formulation and Boundary-Seeded Search

Notation and explanation problem. Let s_t denote the state at timestep t, and let $R_t \in \mathbb{R}^m$ denote the remaining desired-return command. The trained return-conditioned PCN induces action scores (log-probabilities):

$$\ell_\theta(s_t, R_t) = \big(\ell_{\theta,1}(s_t, R_t), \ldots, \ell_{\theta,|\mathcal{A}|}(s_t, R_t)\big)\,. \tag{5.1}$$

Only valid actions are considered. Let $\mathcal{A}(s_t)$ be the valid action set at s_t. Invalid actions are excluded by an action mask that assigns them an effectively negligible

score:

$$\tilde{\ell}_{\theta,a}(s_t, R) = \begin{cases} \ell_{\theta,a}(s_t, R), & a \in \mathcal{A}(s_t), \\ -\infty, & a \notin \mathcal{A}(s_t). \end{cases} \tag{5.2}$$

The original greedy action is:

$$a^\star = \arg\max_{a \in \mathcal{A}(s_t)} \tilde{\ell}_{\theta,a}(s_t, R_t). \tag{5.3}$$

A foil action $a_f \in \mathcal{A}(s_t)$ is selected such that $a_f \neq a^\star$. The explanation problem is to find a perturbation $\delta \in \mathbb{R}^m$ such that the counterfactual command

$$R_{\text{cf}} = R_t + \delta \tag{5.4}$$

causes the foil action to become greedy:

$$a_f = \arg\max_{a \in \mathcal{A}(s_t)} \tilde{\ell}_{\theta,a}(s_t, R_{\text{cf}}). \tag{5.5}$$

State s_t and parameters θ remain fixed: the policy selected $a^\star$ under R_t but would select a_f under R_{cf}.

Feasible command region and normalized perturbations. Counterfactual commands are restricted to a feasible box:

$$\mathcal{C} = \{R \in \mathbb{R}^m : R_{\min} \leq R \leq R_{\max}\}, \tag{5.6}$$

where the inequalities are componentwise and the bounds are the per-objective return ranges fixed in the environment setup. The projection onto this box is:

$$\Pi_{\mathcal{C}}(R)_i = \min\left(\max(R_i, R_{\min,i}), R_{\max,i}\right). \tag{5.7}$$

Because objectives may have different numerical ranges and units, raw perturbation distances are not comparable across objectives. Perturbations are therefore measured in normalized command coordinates. Define:

$$D = \mathrm{diag}(\sigma_1, \ldots, \sigma_m), \qquad \sigma_i = R_{\max,i} - R_{\min,i}. \tag{5.8}$$

If an objective range is zero or numerically invalid, its scale is set to one. Using the column-vector convention of this chapter, the min–max-normalized form of any feasible command R is

$$\bar{R} = D^{-1}(R - R_{\min}). \tag{5.9}$$

For every objective with a positive range, this is componentwise

$$\bar{R}_i = \frac{R_i - R_{\min,i}}{\sigma_i} = \frac{R_i - R_{\min,i}}{R_{\max,i} - R_{\min,i}}. \tag{5.10}$$

Thus $R_{\min,i}$ maps to zero and $R_{\max,i}$ maps to one, while intermediate values retain their relative position within the objective's range.

Apply this normalization to both the current command and the counterfactual command $R_{\mathrm{cf}} = R_t + \delta$:

$$\bar{R}_t = D^{-1}(R_t - R_{\min}), \qquad \bar{R}_{\mathrm{cf}} = D^{-1}(R_t + \delta - R_{\min}). \tag{5.11}$$

The perturbation in normalized command space is their difference:

$$\begin{aligned} z &= \bar{R}_{\mathrm{cf}} - \bar{R}_t && (5.12)\\ &= D^{-1}[(R_t + \delta - R_{\min}) - (R_t - R_{\min})] && (5.13)\\ &= D^{-1}\delta. && (5.14) \end{aligned}$$

The lower-bound offset cancels because it is subtracted from both commands. Equivalently, $\delta = Dz$. The scaled command distance is therefore:

$$d_D(R_{\mathrm{cf}}, R_t) = \left\| D^{-1}(R_{\mathrm{cf}} - R_t) \right\|_2 . \tag{5.15}$$

The projected command associated with a normalized perturbation z is:

$$R(z) = \Pi_{\mathcal{C}} \left(R_t + Dz \right) . \tag{5.16}$$

Thus optimization evaluates only feasible commands.

Target margin. The target margin measures how strongly the foil action is preferred over all other valid actions. Define the strongest non-foil competitor:

$$b_f(R) = \underset{a \in \mathcal{A}(s_t),\, a \neq a_f}{\arg\max} \tilde{\ell}_{\theta,a}(s_t, R). \tag{5.17}$$

The foil margin is:

$$M_f(R) = \tilde{\ell}_{\theta,a_f}(s_t, R) - \tilde{\ell}_{\theta,b_f(R)}(s_t, R). \tag{5.18}$$

The foil is greedy when $M_f(R) \geq 0$. To avoid an unstable boundary solution, a confidence margin $\kappa \geq 0$ requires $M_f(R) \geq \kappa$. The corresponding hinge loss is

$$H_f(R) = \max \left(\kappa - M_f(R), 0 \right) . \tag{5.19}$$

Thus $H_f(R) = 0$ exactly when the foil beats every valid competitor by at least κ.

Counterfactual optimization objective. The black-box counterfactual objective, in normalized coordinates, is:

$$\mathcal{L}(z) = \left\| D^{-1} \left(R(z) - R_t \right) \right\|_2^2 + c\, H_f(R(z)), \tag{5.20}$$

where $c > 0$ weights the targeted flip penalty. The terms penalize normalized command distance and failure to prefer the foil. The objective adapts the Carlini–Wagner targeted margin [59] to ZOO's black-box setting [61]; here the perturbation explains command-dependent action preference rather than inducing adversarial misclassification.

A valid candidate satisfies $M_f(R_{\text{cf}}) \geq \kappa$; the closest valid candidate under d_D is preferred.

Boundary-seeded directional initialization. Local optimization can stall on a flat or non-convex command–action surface, or far from a foil region. Boundary seeding supplies a better initial direction. It adapts Cheng et al.'s direction-and-boundary-distance formulation for hard-label attacks [64], using the foil-greedy command region as the target and the Pareto archive as the directional prior.

Let

$$\mathcal{F} = \{R_1^{\text{front}}, \ldots, R_N^{\text{front}}\} \tag{5.21}$$

be the logged Pareto-front archive of the trained run. Suppose the command originally selected at the beginning of the rollout is R^{des}. At timestep t, the return already collected is:

$$R_t^{\text{col}} = R^{\text{des}} - R_t. \tag{5.22}$$

Each archive point is converted into the remaining command that would still be

required to reach it:

$$\widehat{R}_j = R_j^{\text{front}} - R_t^{\text{col}}. \tag{5.23}$$

Projected residuals serve as outcome-consistent behavioural anchors. The largest-margin anchor,

$$j^\star = \arg\max_j M_f \left(\Pi_{\mathcal{C}}(\widehat{R}_j) \right), \tag{5.24}$$

provides only a coarse direction. Its displacement from the current command,

$$d = \Pi_{\mathcal{C}}(\widehat{R}_{j^\star}) - R_t, \tag{5.25}$$

defines a directional prior: for each objective dimension i,

$$q_i = \begin{cases} +1, & d_i > 0, \\ -1, & d_i < 0, \\ 0, & d_i = 0. \end{cases} \tag{5.26}$$

The search may respectively increase, decrease, or freely vary component i, producing a Pareto-informed cone of directions.

Directional ray search and binary refinement. Candidate directions are sampled in normalized coordinates from a standard Gaussian and normalized to unit length:

$$v \sim \mathcal{N}(0, I_m), \qquad u = \frac{v}{\|v\|_2}, \tag{5.27}$$

then projected to satisfy the sign restrictions defined by q. A command ray is:

$$R(\alpha; u) = \Pi_{\mathcal{C}} \left(R_t + \alpha D u \right), \qquad \alpha \geq 0. \tag{5.28}$$

The maximal step $\alpha_{\max}(u)$ keeps the ray inside the feasible box and sign restrictions. A direction is discarded unless its endpoint satisfies the margin; otherwise the smallest successful step along it is sought:

$$\alpha_f(u) = \inf \{\alpha \in [0, \alpha_{\max}(u)] : M_f(R(\alpha; u)) \geq \kappa\}. \tag{5.29}$$

Binary search over $[0, \alpha_{\max}(u)]$ retains the successful half-interval for $K = 16$ steps, leaving at most $2^{-K} \approx 1.5 \times 10^{-5}$ of the original interval. Since foil regions may be disconnected, this is the nearest success consistent with evaluated midpoints, not a certified global infimum.

The closest successful ray candidate under Eq. (5.15) seeds local refinement:

$$R_{\text{seed}} = \underset{R_{\text{ray}}(u)}{\arg\min}\, d_D(R_{\text{ray}}(u), R_t). \tag{5.30}$$

If none succeeds, local zeroth-order search starts at the original command.

5.2 Zeroth-Order Coordinate Refinement (ZOO-ADAM)

ZOO-style coordinate optimization [61] refines the seed by finite-difference queries to the PCN at fixed s_t, optimizing normalized perturbation z under Eq. (5.20).

At each iteration, a coordinate $i \in \{1, \ldots, m\}$ is sampled uniformly and the partial derivative is estimated by a centered finite difference:

$$\widehat{\nabla_i \mathcal{L}}(z) = \frac{\mathcal{L}(z + he_i) - \mathcal{L}(z - he_i)}{2h}, \tag{5.31}$$

where e_i is the i-th basis vector and $h > 0$ the step. This estimate drives the

original coordinate-wise ZOO-ADAM rule [61]:

$$m_i \leftarrow \beta_1 m_i + (1-\beta_1)\widehat{\nabla_i \mathcal{L}}(z), \tag{5.32}$$

$$v_i \leftarrow \beta_2 v_i + (1-\beta_2)\left[\widehat{\nabla_i \mathcal{L}}(z)\right]^2, \tag{5.33}$$

with bias correction

$$\widehat{m}_i = \frac{m_i}{1-\beta_1^{n_i}}, \qquad \widehat{v}_i = \frac{v_i}{1-\beta_2^{n_i}}, \tag{5.34}$$

where n_i counts the updates applied to coordinate i. The coordinate is then updated as

$$z_i \leftarrow z_i - \eta \frac{\widehat{m}_i}{\sqrt{\widehat{v}_i}+\epsilon}. \tag{5.35}$$

The implementation uses $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\eta = 0.01$, and $h = 10^{-3}$. Each update changes one coordinate and records the closest valid $R(z)$. Only objective evaluations, not PCN gradients, are required.

5.3 White-Box C&W Baseline

The white-box WHITE-CW baseline uses the same state, feasibility box, and validity condition, but optimizes the targeted L_2 Carlini–Wagner attack [59] implemented in the Adversarial Robustness Toolbox [79]. Its classifier maps command perturbation Δ to masked scores $\tilde{\ell}_\theta(s_t, R_t + \Delta)$ for target a_f and optimizes

$$\min_{\Delta} \|\Delta\|_2^2 + c\,\Phi_{a_f}(R_t + \Delta), \tag{5.36}$$

where Φ_{a_f} is the targeted classification loss and the final command is projected to the feasible box, $R_{\mathrm{cf}} = \Pi_{\mathcal{C}}(R_t + \Delta)$.

WHITE-CW differentiates through the policy; CF-ZOO uses only score

queries. Their comparison tests whether gradient access alone suffices, not two methods with identical access.

5.4 Interpretation

Each query is one PCN evaluation at fixed s_t and a candidate R. Anchors cost one query each; directions cost an endpoint query plus 16 queries for a successful binary refinement; and each ZOO-ADAM iteration costs two finite-difference queries plus one updated-command query. The global budgets and measured runtime scaling are reported in Appendix A.

The pair $(s_t, R_t) \mapsto a^\star$ and $(s_t, R_{\text{cf}}) \mapsto a_f$ attributes the action change to the command, not to the environment. Positive ΔR_i asks more of objective i; negative ΔR_i relaxes it. The explanation is a local command-space displacement crossing an action boundary. Because R_{cf} is a PCN input, it is directly issuable, although its rollout need not attain the requested return. This differs from counterfactual states that may be dynamically unreachable [38; 41]: RACCER asks how to reach a state where the foil is chosen [41], whereas this method changes the command at the original state.

The seven RL counterfactual desiderata [38] specialize as follows. *Validity* is $M_f(R_{\text{cf}}) \geq \kappa$; *proximity* is $\|D^{-1}(R_{\text{cf}} - R_t)\|_2$, which keeps the requested trade-off close to the original command; and *actionability* follows because the command is directly issuable. *Sparsity* means changing few objectives, although the present objective minimizes distance rather than ℓ_0. In practice, the explanations are usually sparse, changing 1 or 2 entries. Feasible bounds and Pareto anchors approximate *manifold closeness* without proving it. *Causality* is the local intervention that fixes state and model, while one-step *recourse* reissues R_{cf}.

5.5 Assumptions and Limitations

The following assumptions bound its outputs.

Fixed state and policy; local scope. Holding s_t and θ fixed establishes only an immediate greedy-action change. It does not guarantee a successful trajectory, good termination, or attainment of R_{cf}.

Feasibility versus attainability. A command is *feasible* when every objective lies within its allowed numerical bounds. This does not mean that a trajectory can actually achieve that combination. Being near a logged anchor makes a command more plausible, but is not proof: its objectives may be impossible to achieve together or it may lie in a region rarely seen during training.

Meaningful command use. The method works only when the PCN uses the command at the selected state. If many commands produce almost the same action scores, there may be no nearby action boundary, so the search may fail or require a large change. This can result from a state that strongly favours one action or from representational collapse, where the network ignores its command input [26]. Chapter 7 measures this sensitivity directly.

Failure taxonomy. Failure is ambiguous. The foil may never be greedy inside the box; a valid region may exist but be missed under the finite query budget; the state may dominate the decision; or the required command may lie outside the policy's learned support. Failure therefore does not prove command insensitivity. Chapter 6's grid proves feasibility when it finds a successful command, but a grid can prove impossibility only as a heuristic because a solution may lie between sampled points.

Chapter 6

Counterfactual Experiments

This chapter compares CF-ZOO with a white-box baseline, tests it over a large case set, diagnoses failures, and examines qualitative cases. It uses the eight-environment subset defined in Chapter 3: Branch-Path, Breakable-Bottles, Collect-Two, Deep Sea Treasure, Fruit-Tree, Minecart, Resource-Gathering, and Reward-Line. All evaluated agents are return-only PCNs from Chapter 4. Success rates use Wilson 95% confidence intervals [80]; distances are mean scaled L_2 distances (Eq. (5.15)) over successes.

Each case is a rollout state s_t, remaining command R_t, greedy action $a^\star$, and valid foil $a_f \neq a^\star$. Logged front commands generate greedy rollouts; selected (s_t, R_t) pairs are combined with valid foils. With state and policy fixed, success means finding a feasible R_{cf} that achieves margin κ within budget.

The *stress test* combines all front commands, sampled timesteps, and valid foils. The shared *white-box benchmark* is a subset limited to 12 front commands, early timesteps, and the two highest-probability foils per state. See Appendix A.

6.1 Comparison with the White-Box Baseline

Table 6.1 compares gradient-based WHITE-CW with black-box CF-ZOO on 797 shared cases. Both methods receive the same state, starting command, foil, bounds, and required margin $\kappa = 0.05$; success therefore has the same definition. Their model access differs.

Environment	Method	Cases	Succ.	Rate % [95% CI]	Dist.
Branch-Path	WHITE-CW	35	7	20.0 [10.0, 35.9]	0.270
Branch-Path	CF-ZOO	35	25	71.4 [54.9, 83.7]	0.416
Breakable-Bottles	WHITE-CW	8	2	25.0 [7.1, 59.1]	0.017
Breakable-Bottles	CF-ZOO	8	6	75.0 [40.9, 92.9]	0.030
Collect-Two	WHITE-CW	136	29	21.3 [15.3, 28.9]	0.514
Collect-Two	CF-ZOO	136	136	100.0 [97.3, 100.0]	0.444
Deep Sea Treasure	WHITE-CW	136	14	10.3 [6.2, 16.5]	0.014
Deep Sea Treasure	CF-ZOO	136	50	36.8 [29.1, 45.1]	0.152
Fruit-Tree	WHITE-CW	84	83	98.8 [93.6, 99.8]	0.130
Fruit-Tree	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
Resource-Gathering	WHITE-CW	10	0	0.0 [0.0, 27.8]	–
Resource-Gathering	CF-ZOO	10	4	40.0 [16.8, 68.7]	1.186
Reward-Line	WHITE-CW	148	71	48.0 [40.1, 56.0]	0.380
Reward-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 6.1: Comparison of CF-ZOO with the white-box WHITE-CW baseline on the shared 797-case benchmark. WHITE-CW is the targeted L_2 Carlini–Wagner attack from the ART [59; 79]; it shares cases, bounds, and margin condition but uses gradients rather than a query budget. Success-rate intervals are Wilson score 95% confidence intervals. Dist. is the mean scaled L_2 distance over successful counterfactuals only; failed cases are excluded from distance averages.

WHITE-CW succeeds in 322 cases (40.4%) and CF-ZOO in 637 (79.9%), with a large gap except on Fruit-Tree. Successful WHITE-CW perturbations can be smaller, but its local search often misses distant foil regions: on DST it solves 14 cases at mean distance 0.014, compared with CF-ZOO's 50 at 0.152. Boundary

seeding therefore improves reach, although the methods have different access.

Figure 6.1 shows that ray search seeds most solved cases, ZOO adds successes and reduces many distances, and final binary search removes residual magnitude. The gain over WHITE-CW indicates that Pareto-informed boundary seeding contributes materially.

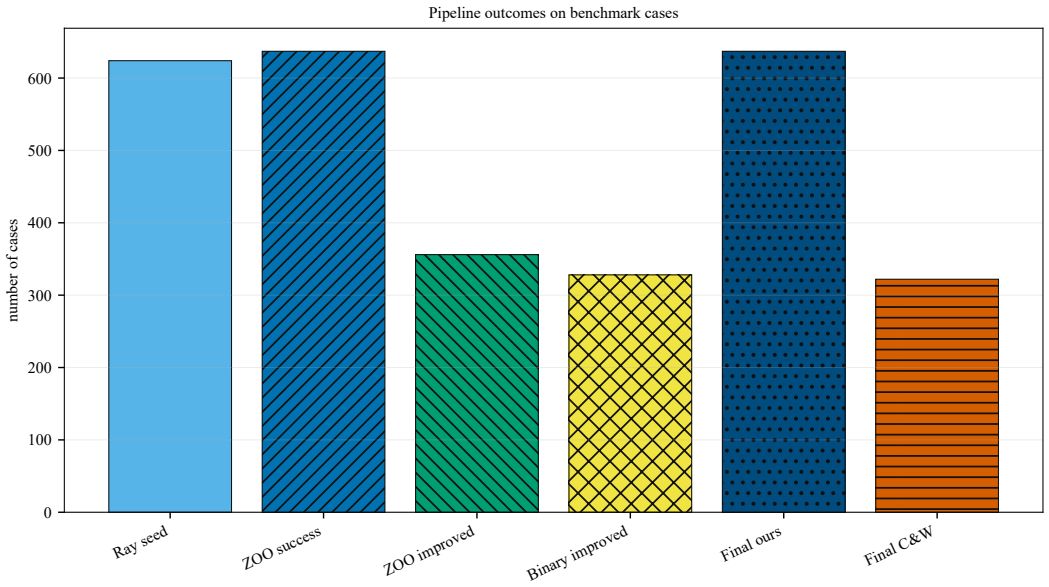


Figure 6.1: Stage outcomes of CF-ZOO on the 797-case benchmark: number of cases with a successful ray seed, cases solved after ZOO refinement, cases in which ZOO improved the seed, cases in which the final binary refinement improved the selected candidate, and the final success counts of CF-ZOO and WHITE-CW.

6.2 Command-Space Decision Landscapes

Figure 6.2 maps $M_f(R)$ over two command dimensions in four cases, fixing state, foil, and other dimensions. Red favours the foil, blue another action, and the contour marks $M_f(R) = \kappa$; markers show both solutions, the original command, and the nearest front anchor. A closer look at the command space helps explain why the two methods behave differently. The figure 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; 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. Using nonlocal rays, CF-ZOO reaches a foil-valid region in all four cases.

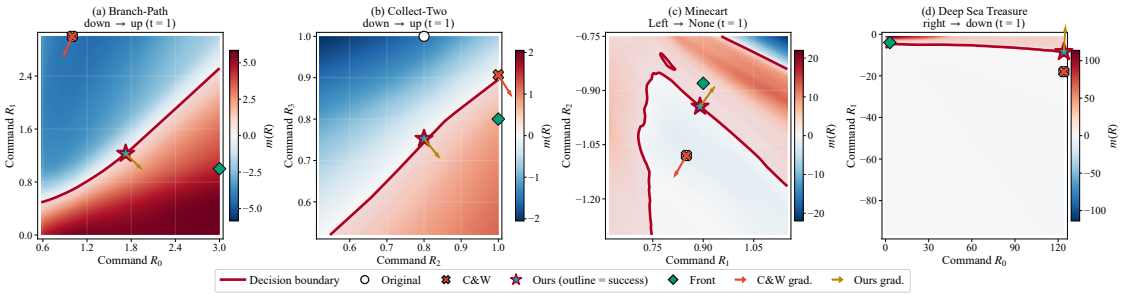


Figure 6.2: Command-space decision landscapes for four qualitative showcase cases (Branch-Path, Collect-Two, Minecart, and Deep Sea Treasure; each panel title states the requested action flip and the rollout timestep). The heat map is the foil margin $M_f(R)$; the red contour is the decision boundary. Markers show the original command (circle), the WHITE-CW solution (cross), the CF-ZOO solution (star; outlined when successful), and the nearest Pareto-front anchor (diamond); arrows show the local margin gradient. Axes are true command values in reward units.

Flat regions and weak gradients trap local optimization. CF-ZOO instead crosses a boundary along a front-informed direction and refines back toward R_t; its solutions lie near learned decision boundaries.

6.3 Combinatorial Stress Test

The stress test covers all available front commands, sampled timesteps, and valid foils, including foils that may be implausible or unreachable (Table 6.2).

Environment	Cases	Succ.	Rate % [95% CI]	Dist.
Branch-Path	44	30	68.2 [53.4, 80.0]	0.466
Breakable-Bottles	8	6	75.0 [40.9, 92.9]	0.030
Collect-Two	152	152	100.0 [97.5, 100.0]	0.493
Deep Sea Treasure	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
Resource-Gathering	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 6.2: Combinatorial stress test of CF-ZOO at the standard budget of 9,001 queries per case. Each case is a state, command, and valid foil action. Success-rate intervals are Wilson score 95% confidence intervals; Dist. is the mean scaled L_2 distance over successful counterfactuals.

CF-ZOO solves 2,422 cases (75.8%). It solves every Collect-Two and Fruit-Tree case and remains effective on continuous-state Minecart. Lower rates on DST and Resource-Gathering may reflect state constraints or unsupported foils, while varying distances show that explanation magnitude is environment-dependent.

6.4 Feasibility Diagnosis

Table 6.3 summarizes the diagnostic outcome categories obtained after rerunning the standard-budget failures with the higher query budget and finite-grid check.

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 6.3: CF-ZOO outcome partition at the 100k diagnostic budget. Feasibility in this diagnostic is established by either a successful search or a valid command on the finite dense grid. The % feasible column is computed over these 2,664 cases (successes plus search-limited failures). † Includes 93 extra cases recovered by raising the budget to 100k. Brackets are Wilson score 95% confidence intervals.

To distinguish optimizer failure from absence of a tested solution, all 773

failures were rerun at a 100,000-query budget and scanned independently on a lattice of 201 values per command dimension (200 subdivisions, abbreviated as a 200-point grid in the paper). If no grid command flips the action, the foil is labelled *grid-infeasible*; this refers only to the finite grid and does not prove infeasibility between its points. A valid grid command missed by the search is labelled a *search-limited failure*.

The high budget recovers 93 additional cases, raising success from 2,422 (75.8%) to 2,515 (78.7%, CI $[77.3, 80.1]$). The grid classifies 531 foils as grid-infeasible (16.6%, CI $[15.4, 18.0]$) and finds valid commands in 149 cases that CF-ZOO still misses, classified as search-limited failures (4.7%, CI $[4.0, 5.5]$). Among the 2,664 cases shown feasible by search or grid, CF-ZOO therefore solves 94.4% (CI $[93.5, 95.2]$), leaving 5.6% search-limited failures. Because the grid is finite, grid-infeasible does not mean formally impossible.

6.5 Qualitative Worked Examples

Branch-Path. 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 $a^\star = \text{up}$. We choose the foil $a_f = \text{down}$. The search returns $R_{\text{cf}} = (0.2953, 1.464)$ and $\delta R = (-2.7047, 0.464)$. 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.

Collect-Two. 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, $a^\star$ = down. We choose a_f = right. The search returns $R_{\text{cf}} = (0, 0, 0.8, 0.8437)$ and $\delta R = (0, 0, 0, -0.1563)$. 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.

Minecart. 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 $a^\star$ = Left. We choose a_f = Right. The search returns $R_{\text{cf}} = (0.3476, 1.22, -0.96)$ and $\delta R = (0.0676, 0, 0)$. 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.

Chapter 7

Goal Influence: How Much Does the Command Matter?

Counterfactuals identify command changes that flip one action but not whether the command matters at that state. Some actions in a rollout may be driven mainly by the state, while others depend strongly on the command. Since a failed search does not show that the command is irrelevant (Section 5.5), this chapter introduces and validates goal-influence measures in ten environments described in Chapter 3. Various tests are performed to assess these methods.

7.1 Confidence and Goal-Influence Measures

A peaked $\pi_\theta(\cdot \mid s_t, R_t)$ may mean either that the state forces one action or that the command decisively selects among alternatives. Entropy, confidence, and margin from one distribution cannot distinguish these cases.

Influence is therefore contrastive across commands: it measures departure from the same policy's generic behaviour at the same state. It is zero when behaviour is command-invariant, regardless of confidence.

Goal-conditioned and broad-generic distributions. Fix a visited state s_t, its valid-action set $\mathcal{A}(s_t)$, and the remaining desired-return command R_t. The trained policy and state remain fixed throughout the calculation. Querying the policy with the command that the agent is currently following gives

$$p_{\text{goal}}(\cdot) = \pi_\theta(\cdot \mid s_t, R_t), \tag{7.1}$$

where the dot denotes the complete vector of actions. Invalid actions are assigned zero probability and the valid probabilities are renormalized to sum to one. Hence, p_{goal} describes what the policy will do under the actual command R_t. One query cannot, however, show whether changing that command would change the decision.

The method therefore queries the same policy at the same state using many generic commands. Let $\mathcal{P}_t$ be the set of unique *probe commands* generated for this decision. Every member of $\mathcal{P}_t$ is a command vector denoted R^{probe} and constructed by Equation (7.3). Their average action distribution is the *broad-generic baseline*

$$p_{\text{base}}(\cdot) = \frac{1}{|\mathcal{P}_t|} \sum_{R^{\text{probe}} \in \mathcal{P}_t} \pi_\theta(\cdot \mid s_t, R^{\text{probe}}), \tag{7.2}$$

where $|\mathcal{P}_t|$ is the number of unique probes. Thus, the R^{probe} in the sum is one generated probe command, not the actual command R_t. For every probe, the policy returns a probability vector while s_t stays unchanged; these vectors are then averaged component by component.

The probe directions cover three kinds of comparison in an m-objective command space. The axes $\pm e_j$ change one objective at a time, where e_j has value one in coordinate j and zero elsewhere. The 2^m sign corners $\{+1, -1\}^m$ change all objectives together under every positive–negative combination; for two objectives these are $(+,+)$, $(+,-)$, $(-,+)$, and $(-,-)$. Finally, the direction of R_t retains

the relative proportions of the current request while discarding its magnitude.

Let $\mathcal{C} = [R_{\min}, R_{\max}]$ be the feasible command box and let $\sigma = R_{\max} - R_{\min}$ contain the numerical range of every objective. Each unit direction u is tested at 50 scale fractions $f \in \{0.02, 0.04, \ldots, 1.00\}$:

$$R_{u,f}^{\text{probe}} = \Pi_{\mathcal{C}}(f \cdot (\sigma \odot u)), \qquad \sigma = R_{\max} - R_{\min}, \tag{7.3}$$

First, $\sigma \odot u$ multiplies each direction component by that objective's feasible range. Next, f chooses between 2% and 100% of that scaled direction. Finally, $\Pi_{\mathcal{C}}$ clips every component to its allowed interval.

Because the construction uses directions and command bounds rather than the size of R_t or the logged front, it requires only black-box policy probabilities and remains independent of the logged-front.

Scores. Goal influence at s_t is the divergence between the goal-conditioned distribution and the broad-generic baseline. The primary signal is the Kullback–Leibler divergence [81], in nats,

$$I_{\text{KL}}(s_t, R_t) = D_{\text{KL}}(p_{\text{goal}} \,\|\, p_{\text{base}}) = \sum_{a \in \mathcal{A}(s_t)} p_{\text{goal}}(a) \log \frac{p_{\text{goal}}(a)}{p_{\text{base}}(a)} \;\in\; [0, \infty). \tag{7.4}$$

This direction weights actions intended under the current command and grows when they are unlikely under the baseline. Three companions reuse the same distributions and probes. Total variation

$$I_{\text{TV}} = \tfrac{1}{2} \sum_{a} |p_{\text{goal}}(a) - p_{\text{base}}(a)| \in [0, 1] \tag{7.5}$$

is the probability mass moved. Jensen–Shannon divergence [82]

$$I_{\mathrm{JS}} = \tfrac{1}{2} D_{\mathrm{KL}}(p_{\mathrm{goal}} \| \bar{p}) + \tfrac{1}{2} D_{\mathrm{KL}}(p_{\mathrm{base}} \| \bar{p}), \qquad \bar{p} = \tfrac{1}{2}(p_{\mathrm{goal}} + p_{\mathrm{base}}), \tag{7.6}$$

is symmetric and bounded by $\ln 2$. Finally, the *probe-flip fraction* asks how often changing the command changes the policy's most probable valid action:

$$I_{\mathrm{flip}} = \frac{1}{|\mathcal{P}_t|} \sum_{R^{\mathrm{probe}} \in \mathcal{P}_t} \mathbf{1}\left[\arg\max_a \pi_\theta(a \mid s_t, R^{\mathrm{probe}}) \neq \arg\max_a \pi_\theta(a \mid s_t, R_t) \right] \tag{7.7}$$

For each probe command, $\arg\max_a$ selects the action assigned the highest probability. The indicator $\mathbf{1}[\cdot]$ equals one when this action differs from the action selected under the actual command R_t, and zero when it is the same. Summing these values and dividing by $|\mathcal{P}_t|$ therefore gives the proportion of probes that change the greedy action. Unlike the averaged baseline, it examines every probe separately and can therefore reveal action changes that averaging may conceal.

Together, the measures describe command influence from two perspectives: KL, TV, and JS quantify how far the current action distribution is from the broad-generic baseline, while I_{flip} records how often alternative commands change the greedy action. They measure sensitivity to the command at a fixed state.

Trajectory profiles. Plotting per-state I_{KL} along a rollout gives a *goal-influence profile* (Figure 7.1). High points in the profile show moments when the command strongly affects the agent's choice. Low points show moments when the current state leads the agent toward the same action regardless of the command. The complete profile therefore reveals whether the goal matters throughout the journey or only at a few important decisions. Despite consistently high confidence, command influence varies by an order of magnitude, distinguishing trade-off-driven from state-forced decisions.

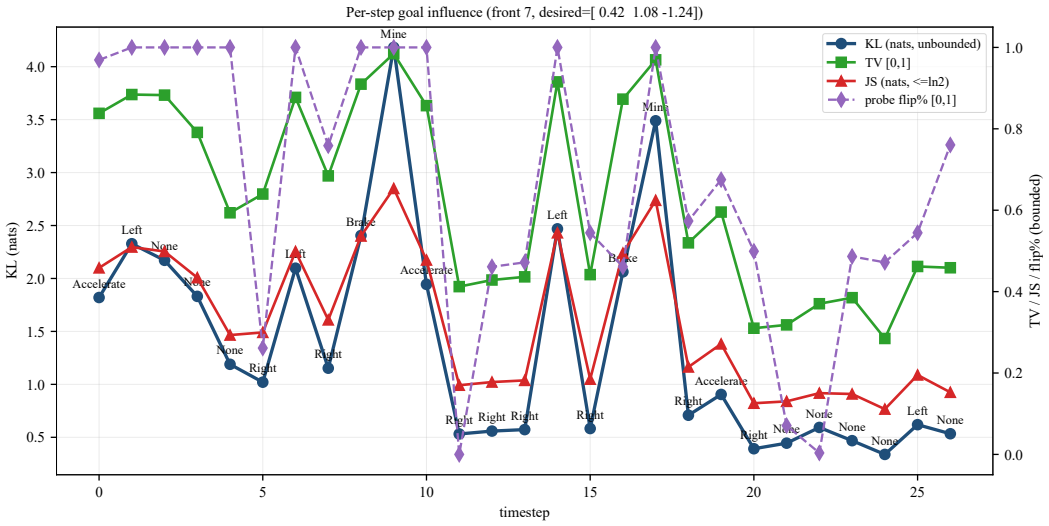


Figure 7.1: Goal-influence profile along the 27-step Minecart rollout of front command 7 (desired return $(0.42, 1.08, -1.24)$). The four signals of Section 7.1 are plotted per timestep, with the greedy action annotated. I_{KL} ranges from 0.34 to 4.18 nats (mean 1.39) along this trajectory and peaks at $t = 9$: the command is decisive at a few steering decisions and near-irrelevant elsewhere, even though the policy is confident throughout.

7.2 Faithfulness Validation

Test are needed. Validation asks whether a high probe-based score really corresponds to stronger behavioural changes when many more commands are tested.

Behavioural ground truth. One approach is to compare with an oracle. At each evaluated state, the state and policy remain fixed while reference commands are tested across the command box. Up to three objectives use a regular grid with 200 values per objective; higher dimensions use 50,000 fixed-seed uniform samples, per dimension. For each reference command, we record the policy's most probable action and compare it with the action chosen under the original command R_t. The *flip fraction* is the proportion of these commands that change the action. It therefore measures how often the policy's decision changes when the command is varied and is used to calculate the ρ_{frac} results in Table 7.1. A

state is *goal-active* when at least one flip exists; this yes–no result provides the labels used by AUROC and, when averaged across states, the Active column.

A useful influence score should agree with these three answers. The symbol ρ denotes Spearman rank correlation, which compares rankings rather than exact numerical values. The column ρ_{frac} checks whether states with higher influence also have larger flip fractions; values closer to 1 are better. AUROC checks whether the score distinguishes goal-active from inactive states [83]; 1 is perfect and 0.5 is chance. It can be interpreted as the chance that a randomly chosen active state receives a higher score than a randomly chosen inactive state, so both kinds of state are required. Minecart, Three-Tree, Collect-Two, and Fruit-Tree contain only active states; without an inactive comparison class, their AUROC is mathematically undefined and reported as *n/a*, not as zero. The column ρ_{near} asks whether an action-changing command is closer when the measured influence is higher. The expected value is negative because the two quantities move in opposite directions: influence rises while the required command distance falls. A value of -1 means this pattern holds perfectly across the ranked states, whereas a value near zero means that higher influence does not consistently correspond to a closer flip. A nearest-flip distance exists only when at least one tested command changes the action, so states with no observed flip are excluded.

Cross-environment results. Table 7.1 summarizes validation on ten environments. Active is the proportion of states where at least one reference command changes the greedy action; for example, 0.82 means that this happens at 82% of the states. The three columns under I_{KL} test the KL score against the GT. The final ρ_{frac} column performs the same ranking test for I_{flip}: it checks whether states that flip under many structured probe commands also flip under many commands in the GT. A value near 1 means that the two methods rank the states similarly.

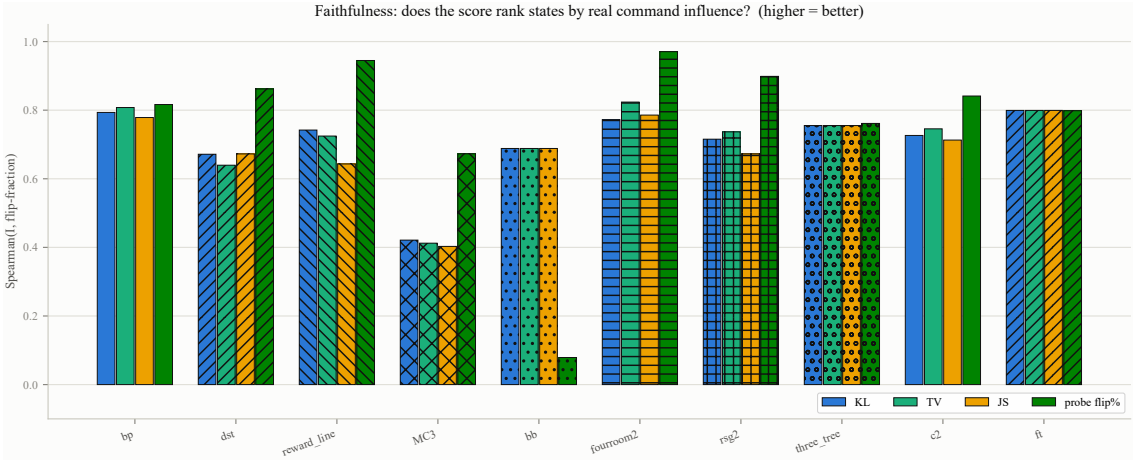
Appendix B reports the complete comprehensive results for all scores.

The results generally support I_{KL}. In nine environments, ρ_{frac} is 0.67–0.80, and the mean across all ten is approximately 0.71. Thus, states with higher KL influence usually change action under more reference commands. Where AUROC can be calculated, it is 0.76–0.98, showing that KL separates active from inactive states better than chance (0.5). Every ρ_{near} is negative, so stronger KL influence usually means that a smaller command change can flip the action.

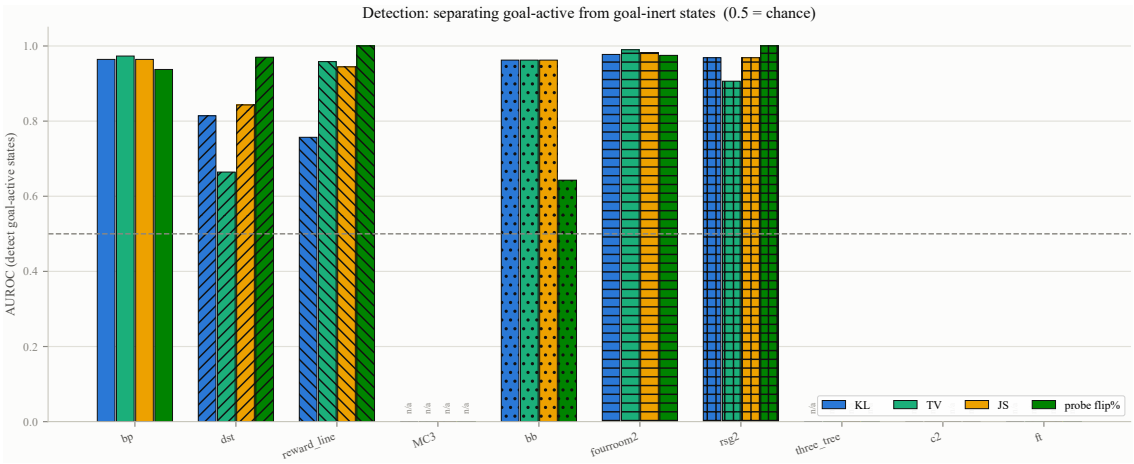
Minecart is the main exception, with $\rho_{\mathrm{frac}} = 0.421$ for I_{KL}. Its action distributions are often *peaked*, meaning that the policy assigns almost all probability to one action. KL has no fixed maximum, so it can become very large when the baseline assigns that action a very small probability. These large values can disturb the ranking of states and weaken the correlation. On Breakable-Bottles, I_{flip} is weak (0.079, compared with 0.688 for KL) because the most probable action rarely changes, even when its probability changes.

						I_{KL} faithfulness			I_{flip}
Env.	m	$\lvert\mathcal{A}\rvert$	States	GT	Active	ρ_{frac}	AUROC	ρ_{near}	ρ_{frac}
Branch-Path	2	4	39	grid	0.82	0.793	0.964	−0.166	0.816
Deep Sea Treasure	2	4	96	grid	0.70	0.672	0.814	−0.372	0.862
Reward-Line	2	4	74	grid	0.97	0.742	0.757	−0.082	0.945
Minecart	3	6	184	grid	1.00	0.421	n/a	−0.662	0.673
Breakable-Bottles	3	3	40	grid	0.53	0.688	0.962	−0.814	0.079
Four-Room	3	4	40	grid	0.50	0.772	0.978	−0.300	0.971
Resource-Gathering	3	4	12	grid	0.67	0.715	0.969	−0.119	0.899
Three-Tree	3	3	56	grid	1.00	0.755	n/a	−0.259	0.761
Collect-Two	4	4	108	MC	1.00	0.726	n/a	−0.685	0.841
Fruit-Tree	6	2	112	MC	1.00	0.799	n/a	−0.174	0.798

Table 7.1: Faithfulness against the dense behavioural reference (GT). m is the number of objectives, $|\mathcal{A}|$ the number of actions, States the number of rollout states evaluated, GT the reference method (grid or Monte Carlo), and Active the proportion of goal-active states. For I_{KL}, ρ_{frac} measures agreement with flip fraction, AUROC measures active-state detection, and ρ_{near} measures association with nearest-flip distance. The final column gives ρ_{frac} for I_{flip}; *n/a* means AUROC cannot be computed because all states are active.



(a) Rank correlation with flip fraction



(b) Detection of goal-active states

Figure 7.2: Faithfulness of the four influence signals. In (a), values closer to 1 mean that a signal ranks states more like the dense-scan flip fraction. In (b), AUROC 1 is perfect detection of goal-active states and the dashed 0.5 line is chance; environments containing only active states are omitted because AUROC requires both classes.

7.3 Observed-Command State Basis

Equation (7.2) compares the action distribution under the current command with an average obtained from structured probes spread across the command box.

Some of those probes may never occur during a rollout. The complementary *state-basis* therefore asks a more practical question: when different Pareto-front rollouts reach the same observed state, does the remaining command change the policy's behaviour?

All visits with the same observation s are placed in one group. Suppose that this group contains n visits with remaining commands $R^{(1)}, \ldots, R^{(n)}$. At visit i, the policy produces the action distribution $p_i = \pi_\theta(\cdot \mid s, R^{(i)})$. The *reachable basis* is their componentwise average,

$$\bar{p}_s(a) = \frac{1}{n} \sum_{i=1}^{n} p_i(a). \tag{7.8}$$

Thus, $\bar{p}_s$ represents the policy's average behaviour at s under the commands that actually occurred there. For each visit, KL divergence measures how far p_i is from this average. The state-basis influence is the mean of these divergences,

$$\mathrm{MI}(s) = \frac{1}{n} \sum_{i=1}^{n} D_{\mathrm{KL}}(p_i \,\|\, \bar{p}_s)\,, \tag{7.9}$$

which, under equal weighting of the recorded visits, is the empirical mutual information between command and action at s. A score near zero means that the observed commands produce similar action distributions. A larger score means that the command gives more information about how the policy behaves at that state. If a state is visited only once, then $\bar{p}_s = p_1$ and the score is zero because no comparison is available. A *fork* is a multi-visit state at which the greedy action changes across observed commands. Forks generally score highly, although a state can also have positive influence when action probabilities change without changing the greedy action. Furthermore, proof of this MI score is required.

We now use the complete action distributions to show why Equation (7.9) is exactly mutual information, relating it to its exact formulation.

Step 1: Assign probabilities to the observed commands. Keep s fixed and select one of its n recorded visits uniformly. Let C be that visit's command, let $\mathcal{R}_s$ contain the distinct commands, and let n_c count the visits using c. Then

$$\begin{aligned} n_c &= \left|\left\{i : R^{(i)} = c\right\}\right|, \\ P(c \mid s) := P(C = c \mid S = s) &= q_s(c) = \frac{n_c}{n}, \qquad \sum_{c \in \mathcal{R}_s} q_s(c) = 1. \end{aligned} \tag{7.10}$$

By Equation (7.10), repeated commands receive proportionally more empirical weight.

Step 2: Define the conditional and joint action probabilities. Let A be an action sampled from the policy. For every valid $a \in \mathcal{A}(s)$,

$$\begin{aligned} P(a \mid c, s) &:= P(A = a \mid C = c, S = s) = \pi_\theta(a \mid s, c) = p_c(a), \\ P(c, a \mid s) &:= P(C = c, A = a \mid S = s) \\ &= P(c \mid s) P(a \mid c, s) = q_s(c) p_c(a). \end{aligned} \tag{7.11}$$

The same action probability is written as $p_i(a)$ when the command is indexed by the visit and as $p_c(a)$ when indexed by the command value.

Step 3: Recover the command free action distribution. Summing the joint probability over all commands removes the command identity:

$$\begin{aligned} P(a \mid s) &:= P(A = a \mid S = s) \\ &= \sum_{c \in \mathcal{R}_s} P(c, a \mid s) = \sum_{c \in \mathcal{R}_s} q_s(c) p_c(a) \\ &= \frac{1}{n} \sum_{c \in \mathcal{R}_s} p_c(a) \qquad \because\ q_s(c) = \frac{1}{n} \\ &= \frac{1}{n} \sum_{i=1}^{n} p_i(a) = \bar{p}_s(a). \end{aligned} \tag{7.12}$$

Thus, $\bar{p}_s$ identifies the policy's action distribution when the command is hidden.

Step 4: Expand the average KL divergence. Group repeated visits by command and use $D_{\mathrm{KL}}(p\|q) = \sum_a p(a)\log\left(\frac{p(a)}{q(a)}\right)$:

$$\begin{aligned}\mathrm{MI}(s) &= \frac{1}{n}\sum_{i=1}^{n} D_{\mathrm{KL}}(p_i \,\|\, \bar{p}_s) \\ &= \sum_{c\in\mathcal{R}_s} q_s(c) D_{\mathrm{KL}}(p_c \,\|\, \bar{p}_s) \\ &= \sum_{c\in\mathcal{R}_s}\sum_{a\in\mathcal{A}(s)} q_s(c)p_c(a)\log\frac{p_c(a)}{\bar{p}_s(a)}.\end{aligned} \tag{7.13}$$

Step 5: Match the formal MI definition and its expectation form. Equations (7.10)–(7.12) imply $q_s(c)p_c(a) = P(c, a \mid s)$ and the ratio

$$\frac{p_c(a)}{\bar{p}_s(a)} = \frac{q_s(c)p_c(a)}{q_s(c)\bar{p}_s(a)} = \frac{P(c, a \mid s)}{P(c \mid s)P(a \mid s)}. \tag{7.14}$$

Substituting Equation (7.14) into Equation (7.13) and using $P(c, a \mid s) = P(c \mid s)P(a \mid c, s)$ gives

$$\begin{aligned}\mathrm{MI}(s) &= \sum_{c\in\mathcal{R}_s}\sum_{a\in\mathcal{A}(s)} P(c, a \mid s)\log\frac{P(c, a \mid s)}{P(c \mid s)P(a \mid s)} \\ &= I(C; A \mid S = s) \\ &= \sum_{c\in\mathcal{R}_s}\sum_{a\in\mathcal{A}(s)} P(c \mid s)P(a \mid c, s)\log\frac{P(c \mid s)P(a \mid c, s)}{P(c \mid s)P(a \mid s)} \\ &= \sum_{c\in\mathcal{R}_s} P(c \mid s)\sum_{a\in\mathcal{A}(s)} P(a \mid c, s)\log\frac{P(a \mid c, s)}{P(a \mid s)} \\ &= \sum_{c\in\mathcal{R}_s} P(c \mid s) D_{\mathrm{KL}}(p_c \,\|\, \bar{p}_s) \\ &= \mathbb{E}_{C|S=s}[D_{\mathrm{KL}}(p_C \,\|\, \bar{p}_s)] \\ &= \frac{1}{n}\sum_{i=1}^{n} D_{\mathrm{KL}}(p_i \,\|\, \bar{p}_s) = \mathrm{MI}(s).\end{aligned} \tag{7.15}$$

Equation (7.15) starts from the standard MI definition at fixed s. The factor $P(c \mid s)$ then cancels inside the logarithm, then the inner action sum becomes a KL divergence and the outer command sum becomes its empirical expectation.

Figure 7.3 shows a clear separation between fork and non-fork states. Mean influence is 0.478 versus 0.050 nats on DST, 0.641 versus 0.021 on Minecart, 0.811 versus approximately zero on Three-Tree, and 0.662 versus approximately zero on Fruit-Tree. Breakable-Bottles and Four-Room each have revisited states but no forks, and their measured state-basis influence is zero. Resource-Gathering has no revisited states, so all of its state scores are single-visit zeros and it is not included in the fork comparison. Blue bars show mean influence at fork states and green bars show it at non-fork states, while each label reports forks divided by multi-visit states. Non-fork influence can be above zero because the greedy action can remain the same even when its probability changes—for example, from 0.90 to 0.60. KL measures this change in the complete action distribution, so it can detect command sensitivity even when no action flip occurs.

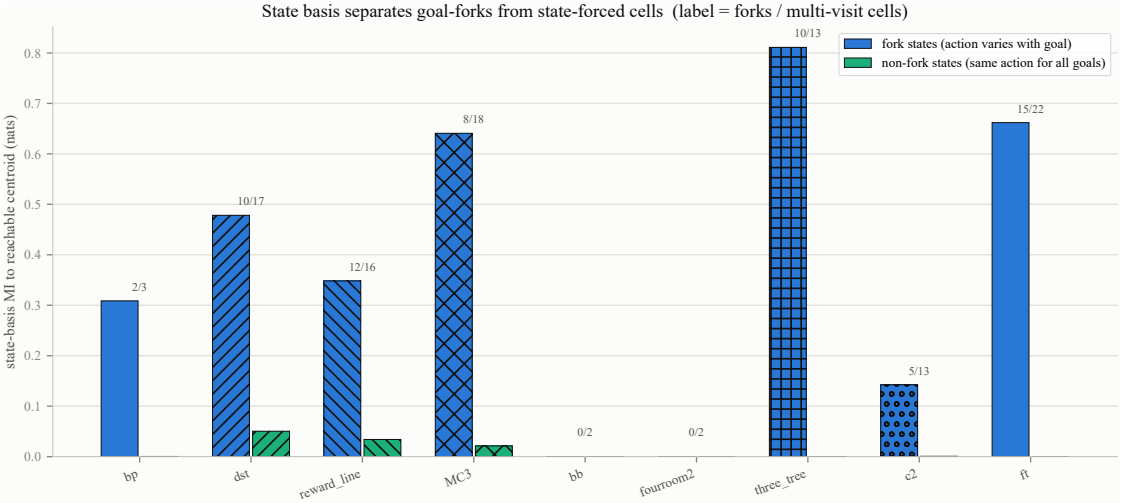


Figure 7.3: Mean reachable-centroid influence (Eq. (7.9)) for fork states, where the greedy action changes across observed commands, and non-fork multi-visit states, where it does not. The label above each environment gives the number of forks over the number of multi-visit states. Breakable-Bottles and Four-Room have no forks and zero measured influence in these results. Resource-Gathering is omitted because it has no multi-visit states.

Chapter 8

An Interactive Explanation and Rollout System

This chapter presents an interactive system for loading a trained PCN, selecting a learned trade-off, inspecting a rollout, requesting a foil explanation, and executing the counterfactual command. The system connects the numerical methods of the preceding chapters to the decision context in which their outputs must be interpreted. A counterfactual vector is more useful when it is shown beside the state that produced it, and a local action flip is more informative when the proposed command can be tested in the environment. The application supports qualitative inspection and future human evaluation.

8.1 Purpose and Requirements

Three requirements shaped the design.

Context. Each counterfactual appears beside the rendered state, action probabilities, remaining command, and collected reward at the queried timestep. This

keeps the explanation tied to one reproducible decision rather than presenting the command change as an isolated vector. It also lets the user relate a change in objective space to the visible situation and to the policy's relative preference for the selected action and foil.

Implementation fidelity. The application vendors the evaluated CF-ZOO, C&W, model, and environment code verbatim, calls the same defaults, and exposes them in settings. Consequently, an interactive query uses the same search logic and parameter meanings as the experiments. Editable settings support inspection without silently creating a separate application-only implementation.

Execution. Because local validity does not ensure return attainment, the verified-realization protocol executes commands and reports when outcomes miss the learned front. The distinction is essential: the counterfactual explains a decision boundary at one state, whereas execution tests a longer sequence of policy–environment interactions. The interface therefore reports both stages instead of treating a successful action flip as evidence that the requested return will be achieved.

8.2 System Design and User Workflow

The application runs entirely on the user's computer from one launch script; it requires no account, external service, or network connection. It has two cooperating parts. A Python JSON server loads the trained model and environment and performs policy inference, environment steps, counterfactual search, replay, and realization checks. An HTML/JavaScript front end provides the controls and displays the rendered state, action probabilities, commands, and results. The browser therefore remains a presentation layer, while all numerical results come

from the same Python implementations used in the experiments.

During a session, the server keeps four related records together: the selected model and logged front, the live environment, the exact rollout history, and the latest counterfactual with its realization attempts. The stored history allows the system to return to the same state and replay the same prefix instead of trying to reconstruct a decision from scratch. Bundled checkpoint models cover the 13 non-Walkroom environments together with Walkroom-2 and Walkroom-3, in both R and RH variants, giving 15 selectable environments. Walkroom-4 through Walkroom-9 are omitted because their large, high-dimensional fronts are difficult to inspect directly in an interactive interface.

An RH model normally receives both a desired return and a remaining horizon. When the user requests a return-command counterfactual, a *horizon-frozen adapter* keeps the horizon at the value used at the queried timestep while CF-ZOO changes only the return. The resulting explanation therefore attributes the action change to the return command rather than to a simultaneous change in the time request.

From the user's perspective, the system follows six steps:

1. **Choose a model.** The model gallery presents R and RH cards for the environments shown in Figure 3.1. The user selects the environment and policy variant before starting, so one session cannot accidentally mix checkpoints. A user may also upload their own trained return-only model or conventional return-plus-horizon PCN agent for a supported environment.

2. **Understand the task.** The environment explainer describes the observation, available actions, rewards, termination rule, and meaning of every objective. This makes each coordinate of the command interpretable; for example, a change in ΔR_j can be linked to objective j rather than read as an unexplained number.

3. **Choose a desired outcome.** The front picker offers returns recorded during training and also accepts a custom return within the command bounds. RH entries additionally contain a horizon. The chosen command remains visible throughout the rollout.

4. **Inspect the rollout.** The user can advance one step at a time, play at 1–4 actions per second, or rewind. The interface shows the current frame, remaining command, collected reward, action probabilities, and action history. Rewind restores the exact recorded prefix, allowing different foils or search settings to be compared at the same numerical state.

5. **Request an explanation.** At any visited state, the user selects a valid foil action and runs CF-ZOO or C&W with visible, editable settings. The result panel reports whether the search succeeded, the counterfactual command R_{cf}, its change ΔR, the foil margin, and the scaled distance. R_{cf} shows the trade-off that produces the foil, while ΔR shows which requested objectives must increase or decrease relative to the current command.

6. **Test the continuation.** The verified-realization protocol executes and animates candidate continuations. This final step separates a command that changes only the next action from one that also finishes near an outcome on the model's logged front.

The complete session can be exported as a text report. It records the model, selected command, queried timestep, foil, explanation measurements, and realization log, allowing the same observation to be reviewed later without relying on screenshots.

8.3 Verified Realization of a Counterfactual Command

A counterfactual is locally valid when R_{cf} makes the policy choose the foil at the queried state. This guarantees only the next action: it does not guarantee that the policy can follow the command for the rest of the episode or achieve its requested return (Section 5.5). The verified-realization protocol therefore restarts from the same recorded decision and tests complete continuations. Each tested continuation is called an *audition*.

Choosing commands to audition. The protocol does not test only the raw R_{cf}. The counterfactual search only needs to cross the current action boundary, so its result may change the first action without describing a stable trajectory learned by the policy. The protocol therefore also creates candidate commands from the checkpoint's logged front. At timestep t, the agent has already collected reward R_t^{col}. Because a logged point R_j^{front} describes the return for a complete episode, the corresponding remaining command is

$$\widetilde{R}_j = \Pi_{\mathcal{C}}\big(R_j^{\text{front}} - R_t^{\text{col}}\big)\,. \tag{8.1}$$

Here $R_j^{\text{front}} - R_t^{\text{col}}$ subtracts what has already been collected, and $\Pi_{\mathcal{C}}$ clips every objective to the allowed command range. For example, if the target total return is $(5, 2)$ and the agent has already collected $(2, 1)$, the remaining command is $(3, 1)$ before clipping.

Duplicate residuals are removed and the rest are tried in order of scaled distance to the current command R_t. The raw R_{cf} is also tested. Finally, a *snap-once* candidate uses R_{cf} for the first decision so that the foil is taken, then switches once to the nearest logged-front residual for the continuation. For RH models,

the logged horizon is reduced by the elapsed timestep and never allowed to fall below one.

Testing and accepting an audition. Each audition starts from the same decision that the user queried. The system first replays the stored actions up to that timestep, restoring the exact state. It then applies the candidate command and checks the policy's next action. If this action is not the requested foil, the candidate is rejected. If it is the foil, the system continues the rollout until the episode ends. After the episode, the achieved total return is compared with every return on the model's logged front. Equation (5.15) gives the scaled distance to each logged point; the smallest value is the audition's final distance, and the corresponding return is its nearest logged-front point. The audition is accepted when this distance is below 10^{-3}.

If no candidate passes, the interface reports the closest attempt but does not label it as successful. For every audition, it shows the candidate command, whether the first action was the foil, the achieved return, the nearest logged-front point and distance, whether the episode ended normally, and an animation of the continuation.

Meaning and limitations. Acceptance establishes two observed facts: the continuation begins with the requested foil, and its final return lies within the stated tolerance of the model's logged front. The first confirms the local counterfactual; the second shows that at least one tested continuation reaches an outcome similar to one represented by the trained policy. The comparison uses the model's logged front, thus access is needed, and only finitely many candidates are tested. In addition, subtracting collected reward from a front point assumes additive returns and is exact only without discounting. Rejection therefore means only that none of the tested continuations passed the two checks.

8.4 Intended Use and Future Evaluation

The testbed supports Section 6.5's examples and renderings and provides a controlled setting for future human evaluation.

Command counterfactuals are deliberately local: ΔR is a sufficient change that crosses the learned policy's action boundary at a fixed state. This scope isolates the effect of the requested trade-off on that decision. A sparse result highlights which objectives need to change while allowing other valid commands. A successful audition strengthens the explanation with rollout evidence by confirming the foil and comparing the realized return with the policy's logged front.

These choices follow five areas for stronger evaluation highlighted by Keane et al. [84]: user studies, plausibility, sparsity, coverage, and comparative testing. This thesis contributes explicit scope, coverage and failure reporting, method comparisons, and rollout checks; the following human-study proposal complements this technical evidence by testing understandability and usefulness.

Human studies are therefore needed, particularly with MORL and RL non-experts. Olson et al. [39] tested whether non-experts could use counterfactual states to identify a flawed RL agent; Amitai et al. [43] tested whether counterfactual action-outcome visualizations improve understanding of agent preferences. Both use human tasks rather than technical scores.

A similar study could ask participants to distinguish well-trained from poorly trained agents in the same environment. A matched study could show non-expert participants the same state, command, and foil under four conditions: no explanation, action probabilities, a command counterfactual, or a counterfactual with rollout evidence (Section 10.2). Participants would predict the action, select a command for the foil, identify the better-trained agent, and rate their confidence. Accuracy, response time, and confident acceptance of failed rollouts may measure usefulness and over-trust. State restoration and session logs would make the comparison reproducible.

Chapter 9

Discussion

This chapter answers the research questions, integrates the contributions, and bounds the claims and validity of the evidence.

9.1 Answers to the Research Questions

RQ1: Can the PCN policy be conditioned on the return alone, and how does this compare with return-plus-horizon conditioning? Yes. With suffix trimming and a shorter-trajectory tie-break, return-only conditioning is broadly comparable to RH across 21 environment configurations: 13 ties, 5 R wins, and 3 RH wins (Table 4.1). RH advantages concentrate in MO-Reacher and 5–8-objective Walkrooms, so the evidence supports neither R nor RH dominance. It supports return as a clean explanatory variable.

RQ2: How can local decisions be explained through minimal command interventions, and how reliably can a black-box search find them? The explanation must identify what should change in the requested objectives, rather than changing the state or retraining the policy. The state and trained PCN are

therefore held fixed while CF-ZOO searches for a nearby desired-return command that makes the specified foil action preferred. The experiments show that this black-box approach can find such changes for most cases in which a solution was observed, without requiring access to policy gradients.

RQ3: How strongly does the command influence the policy at each state, can influence be separated from confidence, and can the signal be validated? Command influence cannot be inferred from confidence alone. Influence is therefore measured by holding the state and policy fixed and comparing the resulting action distributions under different commands. Dense command scans can validate whether high influence corresponds to more frequent action changes. The results generally follow these expected relationships, showing that the measures indeed work.

RQ4: How can command-space explanations be made inspectable and executable? The interactive system combines model and trade-off selection, rewindable rollouts, a clear user interface, and verified realization. It is inspectable research infrastructure, not human-subject evidence.

9.2 Interpretation, Scope, and Validity

Return-only conditioning gives counterfactuals one objective-space intervention. Influence then identifies *where* that intervention matters, while counterfactuals state *what* must change.

Scope of the claims. The evidence establishes comparability, not superiority, of R over RH. Counterfactual validity is an immediate action flip, not attainment, good termination, or Pareto optimality; realization is an empirical test, not a

guarantee. Finite-grid and goal-active classifications are evidence rather than impossibility proofs.

Internal and statistical validity. One seed per environment and generated counterfactual makes the controlled comparison internally consistent but leaves seed variance unknown; narrow wins and ties may change. Wilson intervals quantify case-level uncertainty, not variation between independently trained agents.

Training-instance validity. Every counterfactual and influence score is conditional on the particular trained agent. A newly trained agent given the same environment might act differently. Even with the same architecture, hyperparameters, and training budget, a different model can produce different action probabilities, command–action boundaries, logged fronts, and visited states. A command change that flips the evaluated agent may therefore not flip a retrained agent, and a state that is command-insensitive in one run given a model may be sensitive in another.

Construct validity. Margin validity, scaled L_2 proximity, influence baselines, hypervolume reference points, and the expected-utility weight distribution are defensible choices, not unique constructs; alternatives could change outcomes.

External validity and runtime. Small-to-medium environments use discrete actions, with continuous states only in Minecart and lunar lander. Transfer to continuous actions, visual observations, or substantially more than nine objectives is untested. Timings are implementation-level wall-clock measurements on Appendix A's hardware, not complexity results.

9.3 Positioning Command-Space Counterfactuals within XRL

Building on the mechanisms reviewed in Chapter 2, the central issue here is which explanatory output supplies the evidence needed for a particular decision. The comparison positions command-space counterfactuals by the actor using the explanation, the evidence produced, the required resolution and system access, and their role within XMORL.

Question and answer. A command counterfactual is a preference-sensitivity answer for a person who can set the objective command and encounters a surprising action. It provides an intervention at the existing policy's user-facing control interface. Visual counterfactuals such as Olson et al., GANterfactual-RL, and SAFE-RL naturally support investigation of perceptual evidence [39; 46; 47]. RACCER and ACTER support an operator seeking reachable recourse [41; 42], whereas COUNTERPOL supports a designer considering a changed policy [44]. These methods address different kinds of change.

Type of evidence. Different methods answer different questions. Reward decomposition and trade-off explanations show which rewards or objectives support the current action [49; 50]. The proposed method instead asks how much the requested trade-off must change before the policy chooses the foil. COViz shows what happens after the current and alternative actions [43], while policy-set and trajectory methods summarize broader patterns across policies or trajectories [54; 56]. Their answers may differ because they examine different parts of the decision. For example, the rewards may clearly support an action even though nearby commands do not change it. Conversely, a small command change may flip the action but lead to a poor later outcome. Each method should therefore

be judged by its own purpose: whether the alternative input is plausible, the command change is small, the proposed sequence is reachable, or the reported outcome is actually achieved.

Resolution and access. Command-space explanations trade breadth for a controlled test of one decision. It needs black-box policy queries, a foil, meaningful objective scaling, and feasible command bounds; Pareto anchors additionally seed the search from logged-front returns (Section 5.1). Visual methods need a defensible alternative observation [39; 46; 47]. Outcome and recourse methods typically need rollout or transition access [41; 42; 43], while global summaries need a collection of policies or recorded trajectories [54; 56]. The command intervention is consequently attractive when changing or retraining the policy is not intended.

Role within XMORL. Multi-objective decision support operates at several resolutions. Policy summarization helps navigate broad solution regions [54]; TREX links recurring temporal behaviour to preference trade-offs [56]; and reward or trade-off explanations interpret objective conflict. Command counterfactuals occupy a distinct decision-level role. This can support preference negotiation—for example, identifying how much one requested objective must be relaxed before a chosen foil becomes preferred—which neither a policy cluster nor a recurring trajectory pattern provides by itself.

Practical placement. The methods are best combined only when each resolves a new question. A policy-set summary can help a decision maker understand broad trade-off regions and behavioural trends before examining a particular action [54]. If an action during execution is surprising, a command counterfactual can test whether a nearby preference would change it (Section 5.1). If that

answers the user's concern, no further explanation is needed. If the question becomes what happens after the foil, verified realization (Section 8.3) or COViz [43] adds outcome evidence.

9.4 Reproducibility

The reproducibility package supports both complete experiment runs and inspection of individual reported cases. Together, these materials allow a reader to reconstruct a particular explanation, repeat the main comparisons, and trace summary results back to their underlying records. Appendix A documents the search settings, hardware, and runtime measurements, while Appendix B reports the complete goal-influence results.

The official implementation and reproducibility package are available from the project repository: `https://github.com/JoanikijChulev/Explainable-Multi-Objective-RL-with-PCNs`.

This repository is the primary source for the experiment code, recorded outputs, setup instructions, and executable version of the systems described in this work.

A hosted downloadable online version of the application is available here: `https://joanikijchulev.github.io/CFZOO-Human-Test-App/`.

The hosted interface supports direct exploration of the explanations, whereas the GitHub repository provides the code and experiment artifacts required for technical reproduction.

Chapter 10

Conclusions

10.1 Summary

This thesis asked whether a PCN's desired-return command can serve as its explanatory interface. A return-only variant provides one clean objective-space intervention while remaining broadly comparable to RH across 21 environment configurations (13 ties, 5 R wins, 3 RH wins). CF-ZOO then finds minimal foil-inducing command changes through black-box boundary-seeded search, succeeding in 75.8% of 3,195 cases and 94.4% of cases shown feasible by search or grid. Goal-influence signals distinguish command sensitivity from confidence and track dense-scan behavioural ground truth, while the interactive system makes explanations inspectable and tests their realized continuations against the logged front.

Together, these results establish the command as a practical basis for local, contrastive, objective-level explanation whose influence can be measured, whose action boundaries can be searched, and whose interventions can be executed. The four contributions address successive parts of one explanation problem. Return-only conditioning isolates the variable to be interpreted; influence

analysis identifies states at which that variable affects behaviour; counterfactual search translates an action contrast into a nearby objective-space change; and verified realization tests the proposed intervention beyond its first action. They establish properties of the learned policy, aiding in human understanding of the trained policy.

10.2 Future Work

Human evaluation. Section 8.4 proposes an interactive study in which non-experts use explanations to distinguish a well-trained agent from a poorly trained one. Two smaller studies could test the counterfactual and influence methods directly, without asking participants to judge agent quality. Human testing is important because technical validity alone does not show that an explanation is understood as intended [84].

A first, *counterfactual-comprehension study* could use a short online questionnaire rather than the live testbed. Each static decision card would show a rendered state, plainly named objectives, the current command and action, a foil action, and the returned R_{cf} and ΔR. Participants would see either the numerical change alone or the same values accompanied by a one-sentence explanation. They would identify which objective requests must increase or decrease and choose which conclusion is justified: the foil becomes preferred at this fixed state, not that the whole requested return is guaranteed. Answer accuracy, completion time, and confidence would show whether the explanation communicates both its content and its local scope.

A second, *influence-guidance study* could show participants a short rollout containing several decision states. One version would show only the states and actions; another would add a simple low-to-high command-influence marker de-

rived from the Chapter 7 score. Participants would select the states at which changing the command is most and least likely to change the action. The dense command scan already used for validation would supply the correct comparison. Higher selection accuracy or faster answers with the influence marker would indicate that the score helps people locate preference-sensitive decisions. Unlike the study in Section 8.4, this task evaluates the usefulness of the influence signal rather than an agent assessment.

Sparsity and diversity. The current search minimizes the total scaled L_2 change, so it often changes only a few objectives but does not require this. Future work could add an explicit penalty for the number, or groups, of objectives changed and could return several different counterfactuals with similar distances [33; 34]. A user could then compare alternatives such as relaxing one objective substantially or adjusting two objectives slightly, rather than receiving only one valid trade-off.

Beyond discrete local decisions. This thesis explains a choice between discrete actions at one state. For a continuous-action PCN, the foil would be better expressed as a meaningful action range, such as turning within a specified angle, rather than one exact value [21]. A longer-term extension could instead ask for a trajectory property, such as visiting or avoiding a region. In both cases, the main challenge is to keep the requested alternative clear while still finding a small, understandable command change.

Influence during training. The influence measures are currently applied after training. Computing them during training could reveal when a policy is beginning to ignore its command, a problem related to representational collapse [26]. For example, falling observed-command state-basis influence at a known decision fork

would show that different requested returns are starting to produce the same behaviour. This warning could prompt closer inspection or a training penalty that encourages the policy to remain responsive to its command.

Compressing PCN models. Goal-influence scores could guide an influence-aware compression method. After training the full PCN, a smaller student model could learn one command-independent action pattern at states where different commands consistently produce the same behaviour, while still reproducing the full command-dependent behaviour at influential decision forks. Pruning could then remove unnecessary command-processing capacity. The compressed model would be accepted only if it remains smaller or faster while preserving Pareto performance and command-sensitive decisions.

Certified command robustness. A found counterfactual proves that one tested command changes the action, but an unsuccessful search does not prove that no nearby command can do so. A certification method could instead guarantee that the selected action remains unchanged within a stated region around the current command. Used together, certification would describe the region in which the action is definitely stable, while counterfactual search would provide a concrete command beyond that region at which the action changes.

Appendix A

Hyperparameters, Hardware, and Runtime

A.1 Diagnosis Grid Evaluation

The feasibility diagnosis re-runs failures at 100,000 queries and scans 200 equal subdivisions per objective dimension, including both endpoints. This gives 201 evaluated values per dimension: a two-dimensional command grid contains $201^2 = 40{,}401$ points, while a three-dimensional grid contains $201^3 = 8{,}120{,}601$ points.

Let $\mathcal{G}$ denote this finite command grid. A case is classified as *grid-feasible* when at least one evaluated command satisfies

$$\exists R \in \mathcal{G} \text{ such that } M_f(R) \geq \kappa. \tag{A.1}$$

In other words, the grid found a command for which the foil action beats every other valid action by at least the required margin κ. If no grid point satisfies the condition, the case is labelled grid-infeasible; because the grid is finite, this does not prove that no solution exists between its points.

A.2 Counterfactual Search Settings

Table A.1 lists both experiments' settings; they differ in case generation and CF-ZOO budget, not method constants.

Setting	Stress test	White-box comparison
Search budget		
Query budget per case	9,001	21,000
Candidate directions	9,001	21,000
Binary-refinement steps	16	16
Objective		
Confidence margin κ	0.05	0.05
Flip penalty c	1.0	1.0
ZOO-ADAM		
Learning rate η	0.01	0.01
Finite-difference step h	10^{-3}	10^{-3}
$\beta_1,\ \beta_2$	0.9, 0.999	0.9, 0.999
ϵ	10^{-8}	10^{-8}
Case generation		
Front commands per environment	all	≤ 12
Timesteps per rollout	all sampled	≤ 10
Foils per state	all valid	top 2 by probability
Case cap per environment	—	1,500
Random seed	0	0

Table A.1: Experimental CF-ZOO settings for the CF-ZOO stress test and its evaluation on the shared white-box-comparison cases. Search hyperparameters remain fixed; only the query budget and breadth of the case generation differ.

A.3 Hardware

Experiments used one NVIDIA RTX 4070 and an AMD Ryzen 7 (7000 series). The implementation uses the GPU for neural-network inference and batched policy queries, while environment simulation, search coordination, data processing, and result logging run on the CPU.

A.4 Runtime per Counterfactual

On 797 shared cases, CF-ZOO averages 48.78 s versus WHITE-CW's 8.73 s.

		WHITE-CW (s)		CF-ZOO (s)	
Environment	Cases	Mean	Median	Mean	Median
Branch-Path	35	9.44	9.40	53.21	53.87
Breakable-Bottles	8	8.55	8.30	53.08	53.94
Collect-Two	136	7.98	8.07	45.38	44.91
Deep Sea Treasure	136	8.03	8.10	48.11	48.13
Fruit-Tree	84	10.21	10.09	56.48	61.46
Minecart	240	8.63	8.59	48.71	48.93
Resource-Gathering	10	11.37	11.22	41.28	41.95
Reward-Line	148	9.07	9.06	47.49	47.71
All	797	**8.73**	8.50	**48.78**	48.45

Table A.2: Wall-clock runtime per counterfactual case on the shared 797-case benchmark, at the 21,000-query budget of the white-box comparison.

A.5 Budget Sweep

Table A.3 reports wall-clock time required for one counterfactual search at various query budgets. A budget of 50,000 queries provides the preferred compromise between search results and runtime: four of the ten difficult Minecart cases first succeed at this budget. Absolute runtimes depend on the hardware and implementation and may be reduced by faster hardware or further code optimization.

Query budget	Mean ± SD (s)	Median (s)
10,000	19.1 ± 0.8	19.0
20,000	43.7 ± 1.8	43.5
30,000	60.5 ± 2.5	60.4
50,000	101.4 ± 2.6	102.1
75,000	108.4 ± 3.5	107.3
100,000	176.0 ± 6.4	178.0

Table A.3: Mean (± standard deviation) and median wall-clock time per counterfactual search across maximum query budgets.

Appendix B

Complete Goal-Influence Results

This appendix reports the complete faithfulness and magnitude statistics for all four goal-influence signals (Table B.1).

B.1 Faithfulness of All Four Signals

Probe-flip ranks best in five environments but collapses on Breakable-Bottles ($\rho_{\text{frac}} = 0.079$), where argmax rarely changes. KL, TV, and JS coincide to three decimals in Breakable-Bottles, Three-Tree, and Fruit-Tree. Of the 40 reported ρ_{near} values, 37 are negative; the three positives occur with little distance variance and remain below +0.08. Probe-flip performs best in five environments but fails on Breakable-Bottles because it detects only greedy-action changes, whereas KL, TV, and JS also capture probability shifts. Their matching correlations in three environments indicate identical state rankings, not identical score values. Furthermore, 37 of 40 ρ_{near} values are negative; the three small positives reflect limited distance variation rather than a meaningful reversed relationship.

Environment	Signal	ρ_{frac}	AUROC	ρ_{near}	Mean I	Median I
Branch-Path	KL	0.793	0.964	−0.166	0.338	0.326
	TV	0.808	0.973	−0.194	0.274	0.297
	JS	0.779	0.964	−0.039	0.098	0.103
	Flip	0.816	0.938	−0.346	0.300	0.303
Deep Sea Treasure	KL	0.672	0.814	−0.372	0.304	0.101
	TV	0.640	0.664	−0.425	0.219	0.143
	JS	0.672	0.843	−0.376	0.082	0.028
	Flip	0.862	0.970	−0.501	0.192	0.066
Reward-Line	KL	0.742	0.757	−0.082	0.700	0.595
	TV	0.725	0.958	−0.025	0.445	0.470
	JS	0.643	0.944	+0.072	0.176	0.168
	Flip	0.945	1.000	−0.463	0.554	0.513
Minecart	KL	0.421	n/a	−0.662	1.486	1.071
	TV	0.412	n/a	−0.656	0.646	0.639
	JS	0.403	n/a	−0.665	0.335	0.300
	Flip	0.673	n/a	−0.568	0.641	0.614
Breakable-Bottles	KL	0.688	0.962	−0.814	0.054	0.000
	TV	0.688	0.962	−0.814	0.042	0.000
	JS	0.688	0.962	−0.814	0.017	0.000
	Flip	0.079	0.643	−0.760	0.043	0.000
Four-Room	KL	0.772	0.978	−0.300	0.046	0.011
	TV	0.824	0.990	−0.245	0.094	0.056
	JS	0.786	0.983	−0.179	0.012	0.003
	Flip	0.971	0.975	+0.016	0.117	0.000
Resource-Gathering	KL	0.715	0.969	−0.119	0.051	0.024
	TV	0.737	0.906	−0.262	0.069	0.047
	JS	0.673	0.969	+0.048	0.014	0.008
	Flip	0.899	1.000	−0.667	0.119	0.042
Three-Tree	KL	0.755	n/a	−0.259	1.022	1.050
	TV	0.755	n/a	−0.259	0.633	0.650
	JS	0.755	n/a	−0.259	0.298	0.307
	Flip	0.761	n/a	−0.255	0.633	0.648
Collect-Two	KL	0.726	n/a	−0.685	0.288	0.272
	TV	0.746	n/a	−0.680	0.336	0.352
	JS	0.713	n/a	−0.669	0.070	0.067
	Flip	0.841	n/a	−0.632	0.714	0.915
Fruit-Tree	KL	0.799	n/a	−0.174	1.111	0.664
	TV	0.799	n/a	−0.174	0.507	0.485
	JS	0.799	n/a	−0.174	0.258	0.208
	Flip	0.798	n/a	−0.177	0.507	0.486

Table B.1: Faithfulness of all four goal-influence signals against behavioural ground truth, per environment. ρ_{frac} is the Spearman correlation with the ground-truth flip fraction (higher is better); AUROC is the detection of goal-active states (higher is better), reported as *n/a* where every state is goal-active and the AUROC is undefined for a single class; ρ_{near} is the Spearman correlation with the scaled distance to the nearest action-changing command (negative is better). Mean and median I are in nats for KL and JS, and are unitless fractions in $[0, 1]$ for TV and Flip; magnitudes are therefore comparable within a signal but not across signals.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. 1

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. 1

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2 ed., 2018. 1, 7

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. 1, 7

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. 1, 7

[6] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017. 1

[7] Z. C. Lipton, "The mythos of model interpretability," *Communications of the ACM*, vol. 61, no. 10, pp. 36–43, 2018. 1, 3

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, ACM, 2016. 1

[9] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 4765–4774, Curran Associates, Inc., 2017. 1

[10] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 3319–3328, PMLR, 2017. 1

[11] S. Milani, N. Topin, M. Veloso, and F. Fang, "Explainable reinforcement learning: A survey and comparative review," *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–36, 2024. 2, 3, 10

[12] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013. 2, 8, 9

[13] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, E. Howley, A. A. Irissappane, P. Mannion, A. Nowé, G. Ramos, M. Restelli, P. Vamplew, and D. M. Roijers, "A practical guide to multi-objective reinforcement learning and planning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 1, 2022. 2, 8, 9

[14] R. Dazeley, P. Vamplew, and F. Cruz, "Explainable reinforcement learning for broad-XAI: A conceptual framework and survey," *Neural Computing and Applications*, vol. 35, no. 23, pp. 16893–16916, 2023. 2, 10

[15] M. Reymond, E. Bargiacchi, and A. Nowé, "Pareto conditioned networks," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS '22)*, pp. 1110–1118, IFAAMAS, 2022. 3, 9, 15, 18, 19, 25

[16] D. M. Roijers and S. Whiteson, *Multi-Objective Decision Making*, vol. 11 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool, 2017. 8

[17] H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson, "Multi-objective deep reinforcement learning," *arXiv preprint arXiv:1610.02707*, 2016. 9

[18] J. C. Rosero, N. Cardozo, and I. Dusparic, "Multi-objective deep reinforcement learning optimisation in autonomous systems," in *2024 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pp. 97–102, IEEE, 2024. 9

[19] M. Pirotta, S. Parisi, and M. Restelli, "Multi-objective reinforcement learning with continuous pareto frontier approximation," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, vol. 29, pp. 2928–2934, AAAI Press, 2015. 9

[20] L. N. Alegre, *Sample-Efficient Multi-Task and Multi-Objective Reinforcement Learning by Combining Multiple Behaviors*. PhD thesis, Universidade Federal do Rio Grande do Sul, 2025. 9

[21] V. Slavinskas, "Pareto conditioned networks in continuous action spaces," Master's thesis, Vilnius University, 2024. 10, 70

[22] A. Abels, D. M. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, "Dynamic weights in multi-objective deep reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 11–20, PMLR, 2019. 10, 16, 25

[23] T. Basaklar, S. Gumussoy, and U. Y. Ogras, "Pd-morl: Preference-driven multi-objective reinforcement learning algorithm," in *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. 10

[24] E. Liu, Y.-C. Wu, X. Huang, C. Gao, R.-J. Wang, K. Xue, and C. Qian, "Pareto set learning for multi-objective reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 18789–18797, 2025. 10

[25] Y. Yang, T. Zhou, M. Pechenizkiy, and M. Fang, "Preference controllable reinforcement learning with advanced multi-objective optimization," in *Proceedings of the 42nd International Conference on Machine Learning*, vol. 267 of *Proceedings of Machine Learning Research*, pp. 71612–71647, PMLR, 2025. 10

[26] T. Ambadkar, S. Panda, S. Kale, J. Dodge, and A. Verma, "Preference conditioned multi-objective reinforcement learning: Decomposed, diversity-driven policy optimization," 2026. 10, 35, 70

[27] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," in *Advances in Neural Information Processing Systems*, vol. 32, pp. 14610–14621, Curran Associates, Inc., 2019. 10, 16

[28] E. Puiutta and E. M. S. P. Veith, "Explainable reinforcement learning: A survey," in *Machine Learning and Knowledge Extraction (CD-MAKE 2020)*, Lecture Notes in Computer Science, pp. 77–95, Springer, 2020. 10

[29] G. A. Vouros, "Explainable deep reinforcement learning: State of the art and challenges," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–39, 2022. 10

[30] C. Glanois, P. Weng, M. Zimmer, D. Li, T. Yang, J. Hao, and W. Liu, "A survey on interpretable reinforcement learning," *Machine Learning*, vol. 113, no. 8, pp. 5847–5890, 2024. 10

[31] L. Saulières, *Explaining Reinforcement Learning*. PhD thesis, Université de Toulouse, 2024. 10

[32] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harvard Journal of Law & Technology*, vol. 31, no. 2, pp. 841–887, 2018. 11

[33] S. Dandl, C. Molnar, M. Binder, and B. Bischl, "Multi-objective counterfactual explanations," in *Parallel Problem Solving from Nature (PPSN XVI)*, vol. 12269 of *Lecture Notes in Computer Science*, pp. 448–469, Springer, 2020. 11, 70

[34] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20)*, pp. 607–617, ACM, 2020. 11, 70

[35] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, and C. Shah, "Counterfactual explanations and algorithmic recourses for machine learning: A review," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–42, 2024. 11

[36] A.-H. Karimi, B. Schölkopf, and I. Valera, "Algorithmic recourse: From counterfactual explanations to interventions," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*, pp. 353–362, ACM, 2021. 11

[37] A. Van Looveren and J. Klaise, "Interpretable counterfactual explanations guided by prototypes," in *Machine Learning and Knowledge Discovery in Databases. Research Track (ECML PKDD 2021)*, vol. 12976 of *Lecture Notes in Computer Science*, pp. 650–665, Springer, 2021. 11

[38] J. Gajcin and I. Dusparic, "Redefining counterfactual explanations for reinforcement learning: Overview, challenges and opportunities," *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–33, 2024. 11, 34

[39] M. L. Olson, R. Khanna, L. Neal, F. Li, and W.-K. Wong, "Counterfactual state explanations for reinforcement learning agents via generative deep learning," *Artificial Intelligence*, vol. 295, p. 103455, 2021. 11, 61, 65, 66

[40] S. Tsirtsis, A. De, and M. G. Rodriguez, "Counterfactual explanations in sequential decision making under uncertainty," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 30127–30139, Curran Associates, Inc., 2021. 11

[41] J. Gajcin and I. Dusparic, "Raccer: Towards reachable and certain counterfactual explanations for reinforcement learning," in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS '24)*, pp. 632–640, IFAAMAS, 2024. 11, 34, 65, 66

[42] J. Gajcin and I. Dusparic, "Acter: Diverse and actionable counterfactual sequences for explaining and diagnosing rl policies," *arXiv preprint arXiv:2402.06503*, 2024. 11, 65, 66

[43] Y. Amitai, Y. Septon, and O. Amir, "Explaining reinforcement learning agents through counterfactual action outcomes," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 9, pp. 10003–10011, 2024. 11, 61, 65, 66, 67

[44] S. V. Deshmukh, R. Srivatsan, S. Vijay, J. Subramanian, and C. Agarwal, "Counterfactual explanation policies in rl," *arXiv preprint arXiv:2307.13192*, 2023. Presented at the ICML 2023 Workshop on Counterfactuals in Minds and Machines. 11, 65

[45] S. Dong, S. Zhang, and L. Feng, "Counterfactual explanations for continuous action reinforcement learning," in *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5039–5046, 2025. 11

[46] T. Huber, M. Demmler, S. Mertes, M. L. Olson, and E. 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 (AAMAS '23)*, pp. 1097–1106, IFAAMAS, 2023. 12, 65, 66

[47] A. Samadi, K. Koufos, K. Debattista, and M. Dianati, "Safe-rl: Saliency-aware counterfactual explainer for deep reinforcement learning policies," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 9994–10001, 2024. 12, 65, 66

[48] T. He, J. Gajcin, and I. Dusparic, "Causal counterfactuals for improving the robustness of reinforcement learning," in *AAMAS 2023 Workshop on Autonomous Robots and Multirobot Systems (ARMS)*, 2023. 12

[49] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," in *IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI)*, 2019. 12, 65

[50] R. Sukkerd, R. Simmons, and D. Garlan, "Tradeoff-focused contrastive explanation for mdp planning," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1041–1048, IEEE, 2020. 12, 65

[51] M. Blom, R. Singh, T. Miller, L. Sonenberg, K. Trentelman, A. Saulwick, and S. Wark, "Teaching tactics through multi-objective contrastive explanations," *TechRxiv preprint*, 2024. 12

[52] J. C. Rosero and I. Dusparic, "Explainable multi-objective reinforcement learning: Challenges and considerations," in *Multi-Objective Decision Making Workshop (MODeM 2025)*, 2025. 12

[53] J. C. Rosero, "Explainable multi-objective reinforcement learning," in *ECAI 2025 Doctoral Consortium*, 2025. 12

[54] Z. Osika, J. Zatarain-Salazar, F. A. Oliehoek, and P. K. Murukannaiah, "Navigating trade-offs: Policy summarization for multi-objective reinforcement learning," in *27th European Conference on Artificial Intelligence (ECAI 2024)*, vol. 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 2919–2926, IOS Press, 2024. 12, 65, 66

[55] Q. Xia, T. Wang, and J. M. Herrmann, "Interpretability by design for efficient multi-objective reinforcement learning," *arXiv preprint arXiv:2506.04022*, 2025. 12

[56] D. Rajapakse, J. C. Rosero, and I. Dusparic, "Trex: Trajectory explanations for multi-objective reinforcement learning," *arXiv preprint arXiv:2603.21988*, 2026. Accepted by the 4th World Conference on eXplainable Artificial Intelligence (xAI 2026). 12, 65, 66

[57] P. de las Heras Molins, B. Yalcinkaya, L. Peters, D. Fridovich-Keil, and G. Bakirtzis, "Controllability in preference-conditioned multi-objective reinforcement learning," *arXiv preprint arXiv:2605.10585*, 2026. Accepted at the International Conference on Neuro-Symbolic Systems (NeuS 2026). 12

[58] A. Mone, Z. Osika, F. Felten, P. K. Murukannaiah, M. Fuge, F. A. Oliehoek, and L. C. Siebert, "Objective-behavior alignment: Diagnostics for MORL policy selection," *arXiv preprint arXiv:2606.21321*, 2026. 12

[59] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, IEEE, 2017. 12, 30, 33, 37

[60] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, IEEE, 2016. 12

[61] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. 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 (AISec '17)*, pp. 15–26, ACM, 2017. 12, 30, 32, 33

[62] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992. 13

[63] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *Journal of Machine Learning Research*, vol. 15, no. 27, pp. 949–980, 2014. 13

[64] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019. 13, 30

[65] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *5th International Conference on Learning Representations (ICLR), Workshop Track*, 2017. 13

[66] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020. 13

[67] Y. Faghan, N. Piazza, V. Behzadan, and A. Fathi, "Adversarial attacks on deep algorithmic trading policies," *arXiv preprint arXiv:2010.11388*, 2020. 13

[68] S. Wang, Y. Cao, X. Chen, L. Yao, X. Wang, and Q. Z. Sheng, "Adversarial robustness of deep reinforcement learning based dynamic recommender systems," *Frontiers in Big Data*, vol. 5, 2022. 13

[69] D. Zhang, N. Lashgarian Azad, S. Fischmeister, and S. 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 (ICINCO)*, pp. 665–673, SciTePress, 2023. 13

[70] C.-M. Yu, M.-H. Chen, and H.-T. Lin, "Learning key steps to attack deep reinforcement learning agents," *Machine Learning*, vol. 112, pp. 1499–1522, 2023. 13

[71] F. Felten, L. N. Alegre, A. Nowé, A. L. C. Bazzan, E.-G. Talbi, G. Danoy, and B. C. da Silva, "A toolkit for reliable benchmarking and research in multi-objective reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 36, pp. 23671–23700, Curran Associates, Inc., 2023. 14, 16

[72] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024. 14, 16

[73] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Machine Learning*, vol. 84, no. 1–2, pp. 51–80, 2011. 16

[74] P. Vamplew, C. Foale, R. Dazeley, and A. Bignold, "Potential-based multiobjective reinforcement learning approaches to low-impact agents for AI safety," *Engineering Applications of Artificial Intelligence*, vol. 100, p. 104186, 2021. 16

[75] L. Barrett and S. Narayanan, "Learning all optimal policies with multiple criteria," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pp. 41–47, ACM, 2008. 16

[76] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. 16

[77] L. N. Alegre, A. Bazzan, and B. C. da Silva, "Optimistic linear support and successor features as a basis for optimal policy transfer," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162 of *Proceedings of Machine Learning Research*, pp. 394–413, PMLR, 2022. 16

[78] W. Hung, B.-K. Huang, P.-C. Hsieh, and X. Liu, "Q-Pensieve: Boosting sample efficiency of multi-objective RL through memory sharing of Q-snapshots," in *Proceedings of the 11th International Conference on Learning Representations*, OpenReview.net, 2023. 16

[79] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. M. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.0.0," *arXiv preprint arXiv:1807.01069*, 2018. 33, 37

[80] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927. 36

[81] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951. 45

[82] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991. 46

[83] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. 48

[84] M. T. Keane, E. M. Kenny, E. Delaney, and B. Smyth, "If only we had better counterfactual explanations: Five key deficits to rectify in the evaluation of counterfactual XAI techniques," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 4466–4474, 2021. 61, 69