CRED: Counterfactual Reasoning and Environment Design for Active Preference Learning

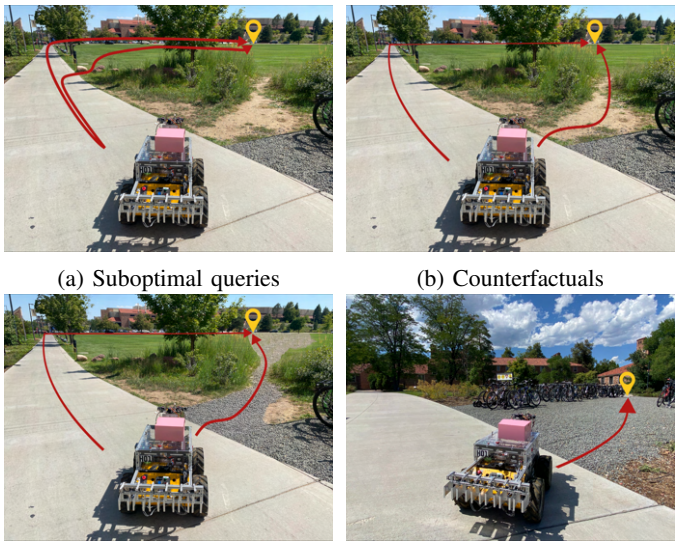
Yi-Shiuan Tung, Bradley Hayes, and Alessandro Roncone
Department of Computer Science, University of Colorado Boulder
{yi-shiuan.tung, bradley.hayes, alessandro.roncone}@colorado.edu

***Abstract*—For effective real-world deployment, robots should adapt to human preferences, such as balancing distance, time, and safety in delivery routing. Active preference learning (APL) learns human reward functions by presenting trajectories for ranking. However, existing methods often struggle to explore the full trajectory space and fail to identify informative queries, particularly in long-horizon tasks. We propose CRED, a trajectory generation method for APL that improves reward estimation by jointly optimizing environment design and trajectory selection. CRED "imagines" new scenarios through environment design and uses counterfactual reasoning–by sampling rewards from its current belief and asking "What if this reward were the true preference?"–to generate a diverse and informative set of trajectories for ranking. Experiments in GridWorld and real-world navigation using OpenStreetMap data show that CRED improves reward learning and generalizes effectively across different environments.**

I. Introduction

Planning under user preferences often involves trade-offs among competing objectives, such as time, cost, and risk. However, users' preferences over these trade-offs are rarely known ahead of time and can vary across individuals and contexts, making it difficult for autonomous systems to make decisions that align with user expectations. One illustrative domain is autonomous delivery, where robots must select routes that balance factors like travel time, energy consumption, tolls, and surface conditions (e.g., paved vs. unpaved) [12, 3]. For example, a user might accept higher energy costs for substantial time savings (e.g., traversing grass) but not for marginal gains. These preferences are hard to hand-specify and vary between different businesses and individuals [28].

Preference learning (PL) aims to learn a reward function based on human preferences between pairs of trajectories, eliminating the need to manually define rewards and allowing for personalization. Unlike inverse reinforcement learning which requires demonstrations, PL allows non-expert users to provide input in complex domains like Atari games, robot locomotion, and routing [8, 14, 27]. However, PL can be sample inefficient, often requiring numerous human preferences for accurate reward learning, which limits its applicability in high-dimensional problems. To address this, Active Preference Learning (APL) finds preference queries—sets of robot trajectories presented to a human—that maximize information gain [21, 4]. Previous work optimizes over a pre-generated set of trajectories [5] or the replay buffer [16] to make optimization tractable, but this method may not sufficiently explore the full feature space, resulting in poor generalization to novel environments, especially for long-horizon problems such as robot routing.



(a) Suboptimal queries (b) Counterfactuals

(c) Environment design (d) Generalization

Fig. 1: The delivery robot above (goal: yellow pin) optimizes its path by considering factors like travel time, energy, safety, and surface conditions. Through active preference learning, it infers human rewards from trajectory rankings. (a) However, current state-of-the-art methods often struggle to efficiently generate informative trajectory pairs for these queries, leading to suboptimal results. To overcome this, our approach incorporates two key contributions: (b) counterfactual reasoning, which explores varied hypothetical preferences to produce more diverse trajectories, and (c) environment design, which "imagines" different scenarios—such as altering terrain from grass to gravel—to enhance the system's generalization capabilities. (d) As a result, the robot can more effectively adhere to human preferences, even in novel environments.

We introduce CRED, a novel and efficient query generation method for APL that learns reward functions which generalize to different scenarios by using 1) **C**ounterfactual **R**easoning and 2) **E**nvironment **D**esign. Instead of generating queries through random rollouts [5, 16], CRED's counterfactual reasoning generates queries that reflect different hypothesized human preferences. Assuming a linear human reward model where $R_H(\xi) = w^T \phi(\xi)$ (the dot product of reward weights

w and trajectory features $\xi(\phi)$), learning the reward function simplifies to learning w. We use Bayesian inference [20] to maintain a belief over w, updated with each human query. By sampling diverse w values (based on cosine similarity) from the current belief, CRED directly evaluates different human preferences and performs counterfactual reasoning, effectively asking "what if w_i or w_j were the true reward weight?" Our second key insight is that the environment influences the generated trajectories and thus the information value of queries. CRED employs Bayesian Optimization [17] to efficiently find environment parameters that yield the most informative preference queries.

Experiments in a GridWorld domain and real-world navigation using OpenStreetMap data [18] demonstrate that CRED generates more informative preference queries and converges faster to the true reward function compared to prior methods. Furthermore, CRED learns rewards that generalize to new environments by querying human preferences in "imagined" scenarios. We assume the robot has knowledge of the features and that ground truth rewards generalize; our goal is to learn the reward weights such that they do generalize. In summary, our contributions are: 1) an environment design approach that enables querying human preferences across diverse contexts, 2) a query generation method that uses counterfactual reasoning which generates trajectories that better reflect different human preferences, and 3) a demonstration of our method's ability to generalize to novel environments through experiments in two domains. Our approach enables robots to learn human preferences more effectively and with fewer iterations.

II. Related Works

Robot Routing. Prior work in autonomous robot routing often employs optimization methods to minimize travel time and costs [19]. However, these methods have difficulty adapting to dynamic environments in real-time due to the high computational cost of the optimization problems, prompting the use of reinforcement learning (RL). Bozanta et al. [6] and Chen et al. [7] apply RL for route planning but focus on rewarding successful deliveries without considering trade-offs between different factors. In contrast, this paper models the reward as a function of several features such as travel time, distance, terrain types etc. Barnes et al. [3] incorporates road properties such as distance, surface condition, and road type, and learns a reward function from a large Google Maps dataset. Our approach, however, focuses on personalization, enabling rapid adaptation of learned rewards by iteratively querying the human for preferences and eliminating the need for large demonstration datasets.

Preference Learning. Preference learning learns the human's reward function by presenting the human with robot trajectories and then asking the human to pick the best one [5]. In active preference learning (APL), the objective is to find the most informative preference query (i.e. a pair of trajectories to query the user) by finding the query that maximizes the expected difference between the prior and the posterior belief distributions over the rewards [21]. Subsequent work improves the objective to maximize the mutual information of the query and the estimated weights which generates queries that are easier for humans to answer [5]. To improve the time efficiency of APL, Biyik and Sadigh [4] proposes batched queries so that queries can be answered in parallel.

A major challenge of APL is generating trajectories that maximize the mutual information objective. Biyik et al. [5] and Lee et al. [16] address this by optimizing within a fixed set of trajectories, but this approach does not scale for long-horizon problems. In addition, the preference queries are generated within the context of the current environment, and previous research has not utilized environment design to improve query quality. Our work uses counterfactual reasoning to generate trajectories based on a belief distribution of reward weights while also optimizing environment parameters to maximize mutual information, thereby improving the information gain of the resulting preference queries.

Environment Design in RL and Robotics. Environment design treats environment parameters as optimizable variables. In RL, it has been leveraged for curriculum learning to improve generalization and convergence, for instance, through co-evolution of agents and environment difficulty [26] or by modifying parameters to maximize an agent's learning potential [10, 1]. In human-robot interaction, environment modification has been used to generate interpretable robot behaviors [15], legible human motion [25], and to support collaborative teaming in settings like warehouse design [29] or tabletop reorganization [2]. Our work applies environment design within APL to find informative preference queries that can effectively reduce the posterior entropy (uncertainty) of the learned reward function.

III. Preliminaries

Model. We consider a fully observable environment modeled as a Markov decision process (MDP) consisting of $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \mathcal{S}_0\}$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $\mathcal{S}_0$ is the initial state distribution. We do not have access to the reward function $\mathcal{R}$ which we have to learn from human preferences. We use $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ to denote the state and action at time t. A trajectory, $\xi \in \Xi$, is a finite sequence of states and actions; $\xi = ((s_t, a_t)_{t=0}^{T})$ where T is the time horizon of the environment.

For autonomous routing, a graph structure is commonly used in which the nodes represent the locations and the edges represent the streets [3]. The set of states $\mathcal{S}$ is therefore the set of nodes, and the actions at each node are the outgoing edges. Our goal is to learn the human's reward function R_H which is modeled as a linear combination of weights w and features Φ of a trajectory, $R_H(\xi) = w^T \Phi(\xi)$. Learning R_H thus simplifies to learning the weights w.

Preference learning. The objective of preference learning is to learn w by querying a human for their preferences between pairs of trajectories. A preference query typically asks "Do you prefer trajectory ξ_A or ξ_B?" [4]. If a human prefers

ξ_A over ξ_B, it implies $R_H(\xi_A) > R_H(\xi_B)$, or equivalently $w^T\Phi(\xi_A) > w^T\Phi(\xi_B)$. From this strict inequality, we can derive that $w^T(\Phi(\xi_A) - \Phi(\xi_B)) > 0$. Let $\psi(\xi_A, \xi_B) = \Phi(\xi_A) - \Phi(\xi_B)$ denote the difference between the features of the two trajectories. The human's preference I can then be encoded by $I = sign(w^T\psi)$.

The human input may be noisy due to uncertainty in their preferences, which can be modeled using Boltzmann rationality, where the likelihood of a preference (Eqn. 1) is determined by a softmax function:

$$P(I \mid \mathbf{w}) = \begin{cases} \frac{\exp(R_H(\xi_A))}{\exp(R_H(\xi_A))+\exp(R_H(\xi_B))} & \text{if } I = +1 \\ \frac{\exp(R_H(\xi_B))}{\exp(R_H(\xi_A))+\exp(R_H(\xi_B))} & \text{if } I = -1 \end{cases} \quad (1)$$

Let $p(w)$ be our current belief distribution of the reward weights. We can perform a Bayesian update to compute the posterior given human input I, $p(w|I) \propto p(I|w)p(w)$. For uniqueness, we constrain the norm of the reward weights such that $\|w\|_2 \leq 1$. Since $p(w)$ can have arbitrary shapes, we use an adaptive Metropolis algorithm [11] to learn the posterior distribution. Based on Biyik et al. [5], the algorithm presents the human with a preference query and updates the belief distribution of w until a fixed number of iterations is reached.

Active Synthesis of Preference Queries. To learn w efficiently using minimal queries, active learning methods select preference queries (ξ_A, ξ_B) that maximize information gain. This is equivalent to maximizing the mutual information between the query and the estimated weights w [5]. Our objective function f is

$$\max_{\xi_A,\xi_B} f(\xi_A, \xi_B) = \max_{\xi_A,\xi_B} H(\mathbf{w}) - \mathbb{E}_I[H(\mathbf{w}|I)] \quad (2)$$

where $H(w) = -\mathbb{E}_w[log(p(w))]$ is the information entropy of the belief $p(w)$. This objective finds preference queries such that the difference between the entropy of the prior and the posterior is maximized. However, evaluating this objective is computationally expensive [4] and often leads to suboptimal solutions. The next section discusses our approach of using counterfactual reasoning and environment design to more effectively generate trajectories that optimize this information gain objective.

IV. Technical Approach

Active preference learning faces challenges in generating trajectories that optimize for information gain (Eqn. 2), as the objective function involves a pair of trajectories as variables. This task is further complicated by the fact that the optimization is typically constrained to a single environment, which may not adequately represent the full feature space, resulting in learned rewards that often fail to generalize effectively. We discuss our approach of using counterfactual reasoning and environment design to address these issues.

A. Counterfactual Reasoning

Counterfactual reasoning explores different trajectories that could result if various reward weights were the true weights. We maintain a belief over the weights while estimating the human's reward function, where each sample of weights could lead to a different policy and consequently different trajectories when the policy is executed. This allows us to pose counterfactual questions, such as "what if reward i is the true reward as opposed to reward j?" Let w_i be an instance of reward weights sampled from our belief. We can train a policy π_i that maximizes the reward function based on w_i by using RL algorithms such as value iteration or PPO [24]. If we rollout the policy π_i in an environment, we get a trajectory ξ_i (Algorithm 1 lines 4-5).

By sampling reward weights and generating trajectories, we construct a set of counterfactual trajectories that represent different human preferences. We then evaluate the information gain objective (Eq. (2)) for each pair of trajectories to identify the most informative preference query (Algorithm 1 lines 7-8). To minimize the number of evaluations of the objective function, we start by sampling N reward weights. We then select the most diverse M weights, where $M < N$, from this set by sequentially computing diversity based on cosine similarity, forming our final set of reward weights for evaluation (Algorithm 1 lines 1-2).

Algorithm 1 Counterfactual Reasoning

Require: Belief $P(w)$, N samples, M subset size
1: Sample $\{w_1, \ldots, w_N\} \sim P(w)$
2: Select M diverse weights (e.g., max cosine distance)
3: **for** each selected w_k **do**
4: Train policy π_k to maximize $w_k^T\Phi(\xi)$
5: Generate trajectory ξ_k by rolling out policy π_k
6: **end for**
7: Compute information gain (Eq. 2) for all pairs ξ_i, ξ_j
8: Return most informative pair (ξ_i, ξ_j)

B. Environment Design

While counterfactual reasoning generates trajectory pairs optimizing for different reward weights, the fixed environment can limit their ability to reveal crucial preference distinctions. We posit that if we have the ability to "imagine" new environments or scenarios, we can better generate trajectories that show the differences between the different reward weights.

More formally, let Θ_E denote the set of environment parameters that the algorithm can modify (e.g. terrain type). The feature function Φ now depends on $\theta_E \in \Theta_E$ and can be represented as Φ_{θ_E}. The return of a trajectory is thus modified as $R_H(\xi) = w^T\Phi_{\theta_E}(\xi)$. Let F be the information gain objective from Eqn. 2 but it includes θ_E as a parameter. We formulate environment design as a bilevel optimization problem:

$$\max_{\theta_E,\xi_A,\xi_B} F(\xi_A, \xi_B, \theta_E) \quad \text{s.t.} \quad (\xi_A, \xi_B) \in \arg\max_{\xi_A,\xi_B} f(\xi_A, \xi_B) \quad (3)$$

The upper level optimization of objective function F selects environment parameters θ_E which is then used by the lower level optimization to find the preference query ξ_A and ξ_B that maximizes information gain f from Eqn. 2.

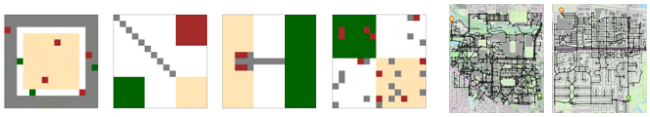
Since $F(\xi_A, \xi_B, \theta_E)$ is generally not differentiable with respect to θ_E, we use Bayesian optimization, a global optimization method that uses a Gaussian process (GP) to model F [23]. GP is a distribution on functions which has a mean function $m : \Theta_E \to \mathbb{R}$ and a positive definite covariance function $K : \Theta_E \times \Theta_E \to \mathbb{R}$. We use the upper confidence bound (UCB) as the acquisition function that selects the next θ_E to evaluate by finding θ_E that maximizes $UCB(\theta_E) = \mu(\theta_E) + \kappa\sigma(\theta_E)$. κ is a hyperparameter that balances exploitation against exploration. Algorithm 2 shows the pseudocode for environment design.

Algorithm 2 Environment Design

Require: Environment parameters Θ_E, Bayesian optimization iterations T

1: **for** $t = 1$ to T **do**
2: Propose θ_E^t using Bayesian optimization
3: Generate (ξ_A, ξ_B) via CR (Algorithm 1) in env θ_E^t
4: Compute information gain $F(\xi_A, \xi_B, \theta_E^t)$
5: Update GP model with $(\theta_E^t, F(\xi_A, \xi_B, \theta_E^t))$
6: **end for**
7: Return optimal θ_E^* found and corresponding (ξ_A, ξ_B)

V. Experiments



(a) GridWorld Navigation (b) OpenStreetMaps

Fig. 2: (a) Sample environments used in the GridWorld Navigation experiments. The terrain types are brick (red), gravel (gray), sand (moccasin), and grass (green). (b) The street graph used in the OpenStreetMap Navigation experiments. Nodes and edges are highlighted in black.

Our experiments aim to address the following hypotheses:

- **H1:** CRED generates queries with higher information gain than those generated by baselines.
- **H2:** CRED learns more accurate rewards with fewer iterations compared to baselines.
- **H3:** CRED learns reward functions that generalize more effectively to novel environments.

A. Environment Setup

GridWorld Navigation. The GridWorld environment includes various terrains, such as brick, gravel, sand, and grass (Fig. 2a). These environments were created by prompting GPT-4 to create realistic scenarios with a diverse distribution of features. From left to right, the environments were titled arid highlands, crossroads pass, coastal village, and forest desert. Our goal is to assess whether our approach effectively learns reward functions that capture the trade-offs between traversing different terrain types. We do not include time and safety as explicit features since they can be inferred from the terrains traversed. Each environment is a 15 x 15 grid where the goal is positioned in the bottom-right corner. We use arid highlands for training and the rest for testing. We also used different environments for training and observed similar results. Thus, we only present the results from training on arid highlands.

The number of environment parameters, $|\theta_E|$, corresponds to the total number of grid cells—225—which is too high for Bayesian optimization (Sec. IV-B) to perform efficiently. To address this, we use variational autoencoders or VAEs [9] to compress the distribution of environments into a lower-dimensional latent space $\mathcal{Z}$. By optimizing over this latent space, we can effectively learn the mutual information objective F (Eqn. 3) as a function of environment parameters.

OpenStreetMap Navigation. We also evaluate CRED on a real-world routing task using data from OpenStreetMaps [18] (Fig. 2b). The features considered include distance, travel time, and elevation changes (ascents/descents). Our experiments focus on last-mile deliveries, restricting the map area to a 500-meter radius from a central latitude and longitude point.

For training, we use a simplified street network consisting of 9 nodes and 12 edges, where distances, travel times, and elevations are sampled uniformly from the ranges [1, 5], [2, 5], and [-1, 1], respectively. In the test environments, these values range from [0.9, 405], [0.1, 291], and [-4.7, 4.7], respectively, though they are heavily skewed towards the lower end. This design ensures that the training differs from the testing environments, making it out of distribution. The environment parameters are the edge features, resulting in a total of 36 variables. While we initially experimented with variational graph autoencoders [13] to optimize over a lower-dimensional latent space, we found that directly using edge features as environment parameters yielded better performance.

B. Baselines

We compare our approach to two state of the art preference learning algorithms. First, we optimize the mutual information objective (Eqn. 2) over pre-generated trajectories obtained from random rollouts [5] which we term **RR**. In addition, we adopt the trajectory generation method from Christiano et al. [8]. We train a policy using the mean of the current belief and generate trajectories through rollouts, allowing the policy to take random actions with probability ϵ (25% in our experiments). We refer to this baseline as the **M**ean **B**elief **P**olicy, abbreviated as **MBP**. Furthermore, we perform ablations of our full system: first, we combine our approach environment design (ED) (Sec. IV-B) with MBP and refer to this as **MBP + ED**. Lastly, we use counterfactual reasoning (Sec. IV-A) alone as a baseline, referring to it as **CR**.

C. Metrics

Belief Entropy. This metric measures the uncertainty of the estimated rewards. The belief over reward weights is estimated by using Monte Carlo Markov Chain which generates likely samples from the distribution. To compute entropy, we first fit

a probability function over these samples using Gaussian kernel density estimation (KDE), which approximates the probability density by using Gaussian kernels as weights [22]. We then create a grid covering the dimensions of the weight vector and compute the probability of each grid cell using KDE. The entropy is approximated as $H \approx \sum_i p(x_i) log p(x_i) \Delta V$ where ΔV is the volume of each grid cell and $p(x_i)$ is the KDE-evaluated density at grid point x_i.

Difference in Rewards. This metric quantifies the percentage difference in cumulative rewards compared to the ground truth. After training, we sample a set of reward weights from the learned belief and train a policy π_{est} for each. We compute the percentage difference in rewards between the estimated policy evaluated under the true reward (R_{est}) and the ground truth policy (R_{gt}): $diff(w_{est}) = (R_{est} - R_{gt})/abs(R_{gt}) * 100$. We report the expected value by averaging over the belief distribution: $\sum_{w_{est}} p(w_{est}) diff(w_{est})$.

Policy Accuracy. This metric quantifies the similarity in action selection between a policy trained on the estimated reward and the ground truth policy. For sampled weights from the belief, we train an estimated policy π_{est}. Accuracy is the proportion of states where the optimal action of π_{est} matches that of the ground truth policy π_{gt}. We report the expected accuracy, weighted by the probability of each sampled weight.

Jaccard Similarity. This metric measures the overlap in visited states between trajectories generated by estimated policies and the ground truth policy. For each estimated policy π_{est}, we generate a trajectory ξ_{est} by executing rollouts with deterministic actions. The Jaccard similarity $J(\xi_{est}, \xi_{gt})$ is the ratio of shared states to the total unique states: $J(\xi_{est}, \xi_{gt}) = |\xi_{est} \cap \xi_{gt}| / |\xi_{est} \cup \xi_{gt}|$. We report the expected Jaccard similarity, averaged over the belief distribution.

VI. Results

Information Gain of Preference Queries. We evaluated our approach using 10 simulated users, each initialized with a different reward weight. This setup demonstrates our method's ability to learn a diverse range of reward functions. To ensure diversity among the ground truth weights, we sample 1000 random weight vectors and select the cluster centers obtained via K-Means as the ground truth weights. Figure 3 shows the belief entropy and information gain objective across training iterations. In both experiments, CRED generates preference queries with higher information gain during the initial 10 iterations. As a result, the entropy of CRED decreases more rapidly in the early stages and ultimately converges to a lower value compared to the baselines, supporting **H1**.

Evaluation of Learned Rewards. Figure 4 shows the box-and-whisker plots illustrating the difference in rewards, policy accuracy, and Jaccard similarity when evaluating the estimated rewards on the test environments. These plots show the distribution of the data, where the central line within each box represents the median, the upper and lower edges correspond to the the first (Q1) and third (Q3) quartiles, and the whiskers extend to 1.5 times the interquartile range from Q1 and Q3. We also plot the mean as black triangles. In

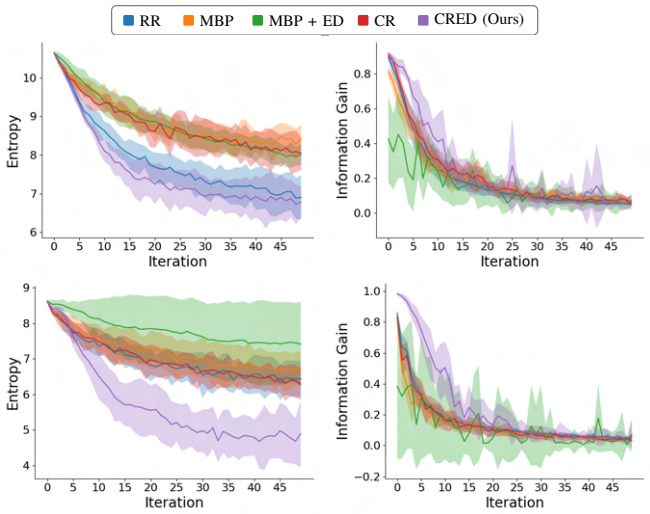


Fig. 3: Belief entropy (left) and information gain from Eq. 2 (right) are shown over the course of training for GridWorld (top) and OpenStreetMaps (bottom). Each line represents the mean, with the shaded region indicating the standard deviation. CRED selects preference queries with higher initial information gain, leading to lower entropy compared to the baselines.

both experiments, CRED reaches the ground truth reward the fastest, converging within 15 iterations, supporting **H2**. The next best-performing algorithm, RR, takes approximately 25 iterations to converge when it does. Its relatively strong performance stems from the lack of a constraint requiring both trajectories in a preference query to reach the goal, allowing it to generate more diverse trajectories in the feature space. Table I shows the mean and standard deviation of test environment metrics at the final training iteration. CRED achieves an 84% and 97% reduction in reward difference, a 5% and 1% absolute increase in policy accuracy, and a 6% and 1% absolute increase in Jaccard similarity compared to the best-performing baseline for GridWorld and OpenStreetMaps, respectively, supporting **H3**. Although the improvements in policy accuracy and Jaccard similarity for OpenStreetMaps are small, it's important to note that the initial values at iteration 0 were already high (90%) due to a strong bias in the reward function towards reaching the goal. Even small differences in these percentages indicate substantial variations in how well preferences are followed.

VII. Conclusion

In this work, we introduce CRED, a novel query generation method for active preference learning that uses counterfactual reasoning and environment design. By sampling from the current belief over reward weights, we pose the counterfactual "what if this reward were the true reward?" These counterfactuals generate trajectories that represent different human preferences. Meanwhile, environment design creates new scenarios to elicit human preferences in varied contexts, enabling the learned reward functions to generalize effectively to novel environments. Through experiments in GridWorld and

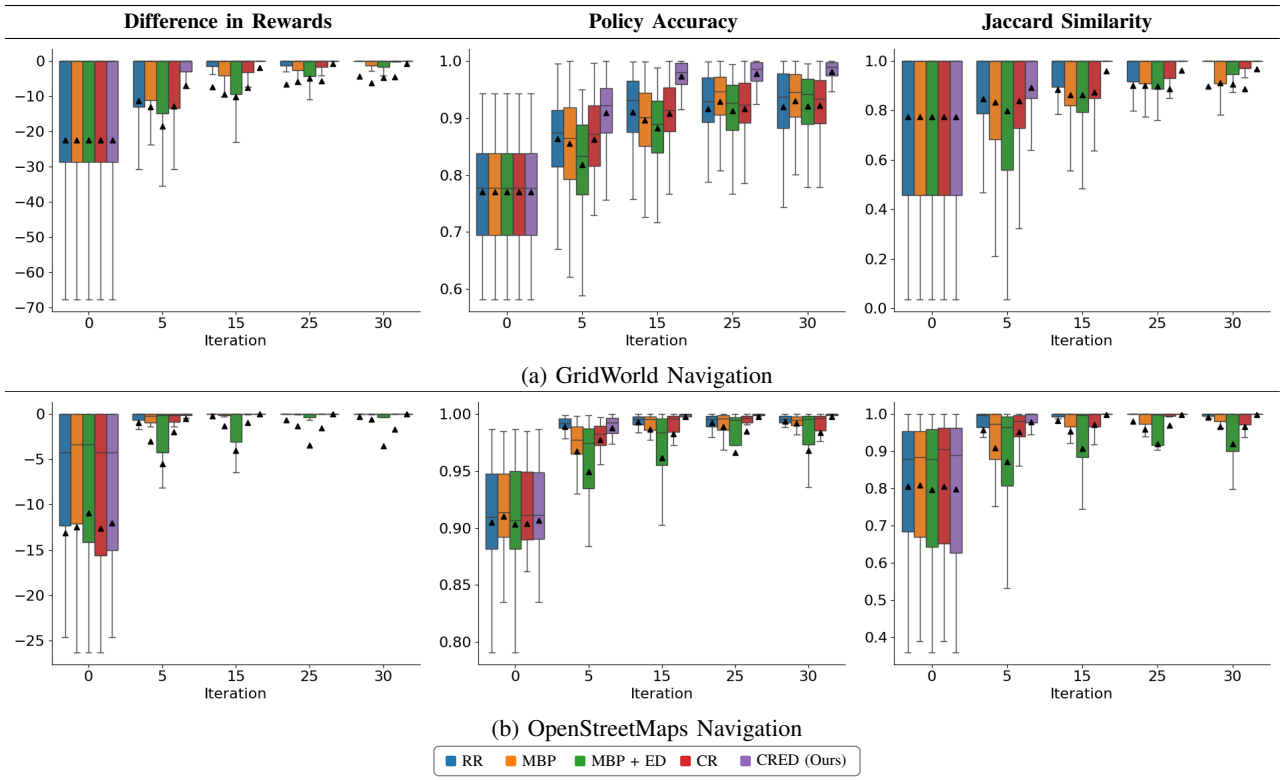


Fig. 4: Box-and-whisker plots depicting the differences in rewards, policy accuracy, and Jaccard similarity when evaluating estimated rewards in the training and testing environments. Black triangles indicate the mean values. CRED converges to the ground truth more quickly and achieves higher performance in test environments.

	GridWorld			OpenStreetMaps		
Condition	**Diff. in Rewards (↑)**	**Policy Accuracy (↑)**	**Jaccard Similarity (↑)**	**Diff. in Rewards (↑)**	**Policy Accuracy (↑)**	**Jaccard Similarity (↑)**
RR	-4.43 ± 13.49	0.92 ± 0.07	0.90 ± 0.22	-0.33 ± 1.28	0.99 ± 0.01	0.99 ± 0.02
MBP	-6.32 ± 17.05	0.93 ± 0.06	0.91 ± 0.18	-0.62 ± 2.35	0.99 ± 0.01	0.97 ± 0.11
MBP + ED	-4.77 ± 15.26	0.92 ± 0.06	0.91 ± 0.20	-3.49 ± 10.62	0.97 ± 0.05	0.92 ± 0.14
CR	-4.54 ± 13.88	0.92 ± 0.07	0.89 ± 0.24	-1.72 ± 7.49	0.98 ± 0.03	0.97 ± 0.07
CRED	**-0.70 ± 6.58***	**0.98 ± 0.02***	**0.97 ± 0.12***	**-0.01 ± 0.05**	**1.00 ± 0.01**	**1.00 ± 0.00**

TABLE I: Evaluation on test environments in the final training iteration. An asterisk (*) denotes statistical significance compared to all baselines based on a one-way ANOVA test.

OpenStreetMaps navigation, we demonstrate that CRED generates preference queries with higher information gain, learns human reward functions in fewer iterations, and achieves better generalization to novel environments compared to previous state of the art methods.

A limitation of our approach is that training a policy using reinforcement learning for the sampled reward weights can be time consuming. Potential solutions include parallelizing training or using meta-learning to develop a single policy that can be quickly fine-tuned for different reward weights.

Acknowledgments

The authors would like to acknowledge Dusty Woods for assistance with figures and image editing.

References

[1] Abdus Salam Azad, Izzeddin Gur, Jasper Emhoff, Nathaniel Alexis, Aleksandra Faust, Pieter Abbeel, and Ion Stoica. Clutr: curriculum learning via unsupervised task representation learning. In *International Conference on Machine Learning*, pages 1361–1395. PMLR, 2023.

[2] Shray Bansal, Rhys Newbury, Wesley Chan, Akansel Cosgun, Aimee Allen, Dana Kulić, Tom Drummond, and Charles Isbell. Supportive actions for manipulation in human-robot coworker teams. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11261–11267. IEEE, 2020.

[3] Matt Barnes, Matthew Abueg, Oliver F Lange, Matt Deeds, Jason Trader, Denali Molitor, Markus Wulfmeier, and Shawn O'Banion. Massively scalable inverse re-

inforcement learning in google maps. In *The Twelfth International Conference on Learning Representations*, 2024.

[4] Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *Conference on robot learning*, pages 519–528. PMLR, 2018.

[5] Erdem Biyik, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, Dorsa Sadigh, et al. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning*, pages 1177–1190. PMLR, 2020.

[6] Aysun Bozanta, Mucahit Cevik, Can Kavaklioglu, Eray M Kavuk, Ayse Tosun, Sibel B Sonuc, Alper Duranel, and Ayse Basar. Courier routing and assignment for food delivery service using reinforcement learning. *Computers & Industrial Engineering*, 164:107871, 2022.

[7] Xiaoyang Chen, Yunxiang Gan, and Shuguang Xiong. Optimization of mobile robot delivery system based on deep learning. *Journal of Computer Science Research*, 6 (4):51–65, 2024.

[8] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[9] Lucas Pinheiro Cinelli, Matheus Araújo Marins, Eduardo Antonio Barros Da Silva, and Sérgio Lima Netto. *Variational methods for machine learning with applications to deep networks*, volume 15. Springer, 2021.

[10] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 2020.

[11] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223 – 242, 2001.

[12] Mokter Hossain. Autonomous delivery robots: A literature review. *IEEE Engineering Management Review*, 51 (4):77–89, 2023. doi: 10.1109/EMR.2023.3304848.

[13] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[14] W Bradley Knox and Peter Stone. Augmenting reinforcement learning with human feedback. In *ICML 2011 Workshop on New Developments in Imitation Learning (July 2011)*, volume 855, 2011.

[15] Anagha Kulkarni, Sarath Sreedharan, Sarah Keren, Tathagata Chakraborti, David E Smith, and Subbarao Kambhampati. Designing environments conducive to interpretable robot behavior. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10982–10989. IEEE, 2020.

[16] Kimin Lee, Laura M Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, pages 6152–6163. PMLR, 2021.

[17] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014. URL https://github.com/bayesian-optimization/BayesianOptimization.

[18] OpenStreetMaps. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org, 2017.

[19] Manuel Ostermeier, Andreas Heimfarth, and Alexander Hübner. The multi-vehicle truck-and-robot routing problem for last-mile delivery. *European Journal of Operational Research*, 310(2):680–697, 2023.

[20] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[21] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

[22] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[23] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[24] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[25] Yi-Shiuan Tung, Matthew B Luebbers, Alessandro Roncone, and Bradley Hayes. Workspace optimization techniques to improve prediction of human motion during human-robot collaboration. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 743–751, 2024.

[26] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

[27] Nils Wilde, Alexandru Blidaru, Stephen L Smith, and Dana Kulić. Improving user specifications for robot behavior through active preference learning: Framework and evaluation. *The International Journal of Robotics Research*, 39(6):651–667, 2020. doi: 10.1177/0278364920910802. URL https://doi.org/10.1177/0278364920910802.

[28] Bin Yang, Chenjuan Guo, Yu Ma, and Christian S Jensen. Toward personalized, context-aware routing. *The VLDB Journal*, 24:297–318, 2015.

[29] Yulun Zhang, Matthew C. Fontaine, Varun Bhatt, Stefanos Nikolaidis, and Jiaoyang Li. Multi-robot coordination and layout design for automated warehousing. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 5503–5511. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/611. URL https://doi.org/10.24963/ijcai.2023/611. Main Track.