Interactive Reward Tuning: Interactive Visualization for Preference Elicitation

Danqing Shi[1], Shibei Zhu[1], Tino Weinkauf[2], Antti Oulasvirta[1]

Abstract— **In reinforcement learning, tuning reward weights in the reward function is necessary to align behavior with user preferences. However, current approaches, which use pairwise comparisons for preference elicitation, are inefficient, because they miss much of the human ability to explore and judge groups of candidate solutions. The paper presents a novel visualization-based approach that better exploits the user's ability to quickly recognize interesting directions for reward tuning. It breaks down the tuning problem by using the visual information-seeking principle: overview first, zoom and filter, then details-on-demand. Following this principle, we built a visualization system comprising two interactively linked views: 1) an embedding view showing a contextual overview of all sampled behaviors and 2) a sample view displaying selected behaviors and visualizations of the detailed time-series data. A user can efficiently explore large sets of samples by iterating between these two views. The paper demonstrates that the proposed approach is capable of tuning rewards for challenging behaviors. The simulation-based evaluation shows that the system can reach optimal solutions with fewer queries relative to baselines.**

I. INTRODUCTION

Reward design [1] is a fundamental process in reinforcement learning (RL) that has a direct impact on the behavior of the RL agent. Designing a satisfactory reward function to guide the agent's learning process involves several steps such as reward term specification, reward shaping, and reward tuning. Designing a reward function can be challenging since expressing the task's nature often requires domain knowledge and lots of experimentation. A reward function that seems correct to the reward's designer might lead to unexpected behaviors by the RL agent, such as failure at the task [2] or myopic behavior with partial completion of the task when it involves a long task horizon [3]. While reward terms can be easily specified via domain knowledge, designing a combination that is perfect for reaching the desired behavior remains challenging. The agent might exhibit unstable behavior upon even tiny changes in reward terms [2], [4], yet there have been few studies of the intricate relations between multiple objectives and the resulting behavior [5]. Despite the critical role of the reward function, its designing process is often done by trial and error [6], where an expert proposes a reward function, inspects the resulting agent's behavior, proposes changes, and re-iterates the process.

In this paper, we tackle the problem of *reward tuning* [7] from humans' feedback, which involves seeking a reward function that produces user-desirable behavior by using human feedback. This process is commonly framed as *preference elicitation*. Existing preference-based methods that learn from human feedback [8]–[10] engage users in a tedious feedback process with pairwise comparisons or ranking, which can be time-consuming (Fig. 1-a). While these methods have shown success, they are often impractical for real-world applications. Take the example of the most popular method: pairwise comparison. It inherits limitations in dealing with user preferences that can be improved:

- Users have limited control over the queries presented to them. Even though active query strategies can minimize the number of preference queries [11], they still require users to assess uninteresting query instances repeatedly.
- The interface for pairwise-comparison queries is ineffective in that it only displays instances side by side, without giving users any other contextual information.
- Pairwise comparison misses out on an opportunity for preference elicitation to factor in user expertise, which could accelerate the process.

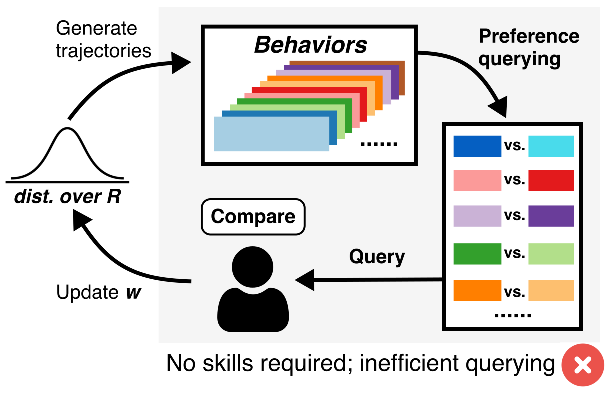
To solve the problems mentioned above, we introduce an approach for interactive visualization-based feedback that supports users' decision-making. Interactive interfaces have been proven to improve the user's ability to explore the design space [12]. To the best of our knowledge, our system is the first to support an inspection mechanism with information for visual analytics, thus allowing better user feedback.

We illustrate the concept of our method in Fig. 1-b, which follows the visual-information-seeking principle: *overview first, zoom and filter, then details-on-demand* [13]. To make it easier to explore policies, the system projects the trajectories generated from the policies into an embedding space. Users can freely explore this space and select specific behaviors of interest. Once selected, the system provides a detailed sample view that displays the selected behaviors and relevant analysis data. Users can analyze this information in detail and decide which behavior to inspect next. With our system, we alleviate the limitations mentioned above leading to:

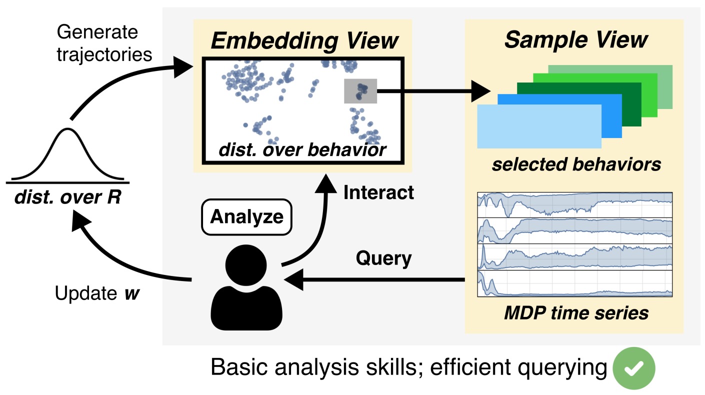
- Users can comprehend the behavior space by examining groups of sampled behaviors. This allows users to develop a good understanding of the behavior after a few observations.
- Given the interactive inspection mechanism, users can influence the data points to be presented by interacting with the system. Unlike the current approaches that rely on active learning methods for query selection, instead of asking for preferences regarding the comparison of

[1]Danqing Shi, Shibei Zhu and Antti Oulasvirta are with Aalto University, Finland `firstname.lastname@aalto.fi`

[2]Tino Weinkauf is with KTH Royal Institute of Technology, Sweden `weinkauf@kth.se`



(a) Pairwise comparison



(b) Interactive reward tuning

Fig. 1: We present an interactive reward tuning approach to better utilize human experts' knowledge and abilities in reward tuning. (a) Previous methods that use pairwise comparisons for preference elicitation can be inefficient. Users must perform numerous comparisons to find the optimum. (b) To address this issue, our approach uses a visualization system to decompose the preference elicitation via two linked views: an Embedding View and a Sample View. The Embedding View visualizes the distribution of behaviors in a two-dimensional space for efficient clustering and querying, while the Sample View displays both selected behaviors and the Markov decision process (MDP) time series that can support the visual analysis. The system enables users to iteratively switch and analyze from two views, efficiently tuning rewards with fewer queries. Using rational designer models, we prove that interactive reward tuning is more efficient than pairwise comparison.

samples selected via active learning, users can actively direct the exploration within the sample space and focus on the most promising area.

- The visual information and the data analysis of the selected behaviors can support the user's decision-making process leading to faster convergence of the desired policies with less number of queries.

We conduct a simulation-based study to evaluate and compare the performance of our approach with the baselines under two user models, including the noisy model and Boltzmann rationality [14], [15]. The results indicate that our approach requires 15 queries to achieve an average utility of 0.95 and 0.97 in two user models, outperforming the standard pairwise comparison by 2.6% and 7.5%.

The main contributions of this paper are:

- A novel interactive visualization system to improve the efficiency of reward tuning;
- A preference elicitation workflow that integrates the visualization system to train a satisfactory policy;
- A simulation-based evaluation through modeling rational users demonstrates the efficiency of our approach relative to baseline methods.

II. Background

A. Markov Decision Process

In reinforcement learning, a sequential decision-making process can be represented by a Markov decision process (MDP) [6], defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$ where $\mathcal{S}$ is the state space and $\mathcal{A}$ the action space; $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the state-transition possibility of a given action. $R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The optimal policy results from optimizing the expected state–action value function $\pi^*(a|s) = \max_\pi Q^\pi(s, a)$, where $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^t R_{t+k+1} | s_t = s, a_t = a\right]$.

B. Reward Tuning Problem

The goal of reward tuning is to adjust a reward function to achieve the user-desired policy for the agent. The reward function is often assumed to be a linear combination of various state and action features [8]:

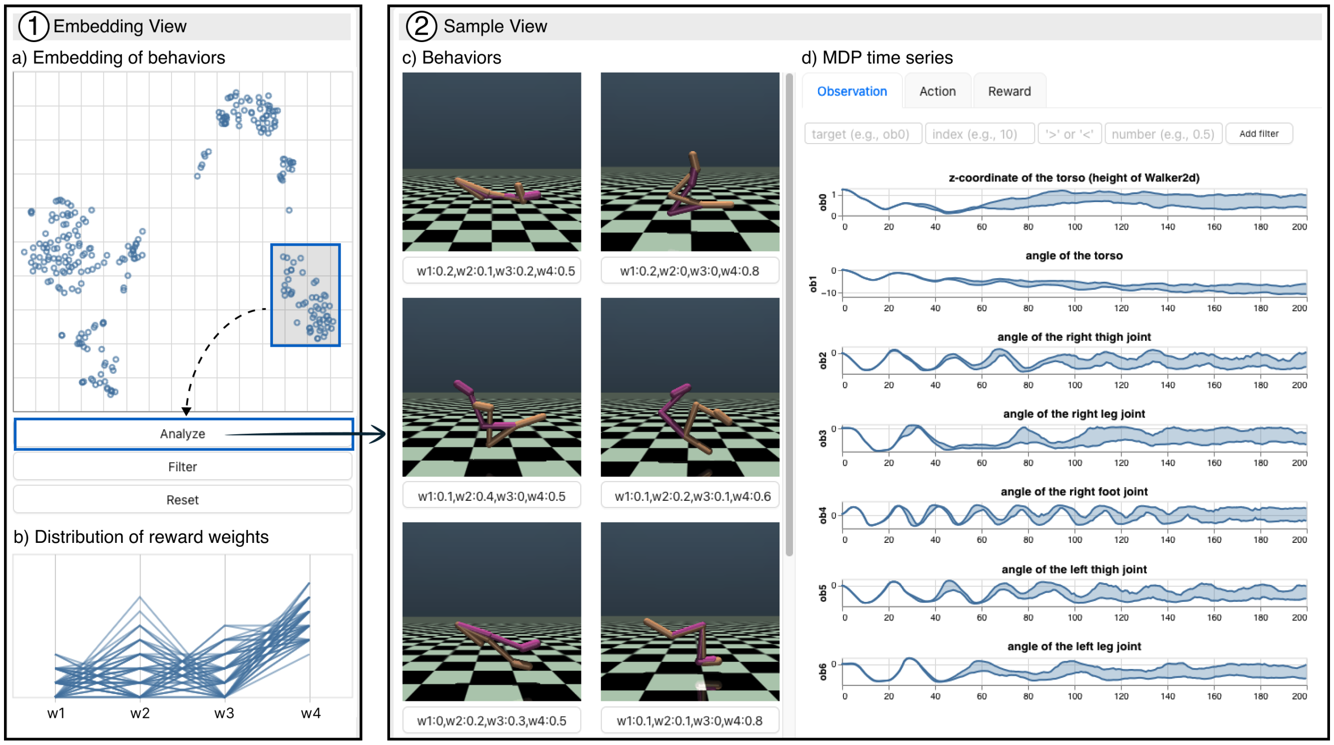
$$r(s, a) = \sum_{i=0}^{d} w_i \phi_i(s, a) = \boldsymbol{w}^\top \Phi(s, a) \tag{1}$$

where $\Phi(\cdot)$ is the feature vector and $\boldsymbol{w}$ is the weight vector, $\boldsymbol{w} \in \mathbb{R}^d$, with $\|\boldsymbol{w}\|_2 \leq 1$. The process of reward tuning consists of adjusting the $\boldsymbol{w}$ value in the expression.

C. Reward Tuning via Preference Elicitation

Reward tuning via preference elicitation is the process of learning a user's preferences from among a set of options by gathering user feedback [9]. Typically, users are asked to express their preferences using relative feedback since providing this in the form of statements such as "I prefer A over B" [16] is easy. Therefore, preference elicitation is usually based on pairwise comparison, for which a user query is defined as $\mathcal{Q} = \{(\tau_i, \tau_j; \mathrm{o})\}$, where τ_i, τ_j are the state–action trajectories that result from a policy and $\mathrm{o} = \{\prec, \succ, \sim\}$ is the preference-order operator specifying the order relationship between the two trajectories. Given the estimated expected return $\hat{R}$ for the choice options τ_i, the preference order is commonly defined thus:

$$\begin{aligned} p(\tau_i \succ \tau_j|\boldsymbol{w}) &= \mathrm{I}(|\hat{R}(\tau_i|\boldsymbol{w}) - \hat{R}(\tau_j|\boldsymbol{w})| > \epsilon) \\ p(\tau_i \prec \tau_j|\boldsymbol{w}) &= \mathrm{I}(|\hat{R}(\tau_j|\boldsymbol{w}) - \hat{R}(\tau_i|\boldsymbol{w})| > \epsilon) \\ p(\tau_i \sim \tau_j|\boldsymbol{w}) &= \mathrm{I}(|\hat{R}(\tau_i|\boldsymbol{w}) - \hat{R}(\tau_j|\boldsymbol{w})| \leq \epsilon) \end{aligned} \tag{2}$$



1 Embedding View
a) Embedding of behaviors
Analyze
Filter
Reset
b) Distribution of reward weights
w1
w2
w3
w4
2 Sample View
c) Behaviors
w1:0.2,w2:0.1,w3:0.2,w4:0.5
w1:0.2,w2:0,w3:0,w4:0.8
w1:0.1,w2:0.4,w3:0,w4:0.5
w1:0.1,w2:0.2,w3:0.1,w4:0.6
w1:0,w2:0.2,w3:0.3,w4:0.5
w1:0.1,w2:0.1,w3:0,w4:0.8
d) MDP time series
Observation
Action
Reward
target (e.g., ob0)
index (e.g., 10)
'>' or '<'
number (e.g., 0.5)
Add filter
z-coordinate of the torso (height of Walker2d)
angle of the torso
angle of the right thigh joint
angle of the right leg joint
angle of the right foot joint
angle of the left thigh joint
angle of the left leg joint

Fig. 2: The user interface of the visualization system presents two views: the Embedding View (1) and Sample View (2). These mutually linked views support exploring the agent's behavior space in a controlled manner. The screenshots illustrate a user tuning the reward for the agent's backflip. In the Embedding View, a scatterplot lets the user observe the distribution of the behaviors generated and inspect a cluster by brushing the points (a). Based on the selection, a corresponding distribution of reward weights is displayed for the parallel coordinates (b). Clicking the "Analyze" button displays behavior videos in the Sample View (c). In addition, the system presents detail-level observation, action, and reward data through MDP time-series charts (d). The user can analyze the behaviors and use the filter feature at the top to perform further inspection.

where I is the indicator function and ϵ is a threshold value that determines the random noise level. The value of ϵ is directly affected by the cognitive capability of the human users. That is, given the current choice set of the query, the human's internal evaluation of the expected return $R(\cdot)$ over the choices is subject to some level of noise. In addition, while the optimal policy is assumed to entail always maximizing the true reward function, human users are not always optimal. To reflect this, the user's decision can be modeled via a Boltzmann rational agent [14], [15]:

$$p(\pi_\theta(a|s,\boldsymbol{w})) = \frac{\exp \beta Q^\pi(s,a|\boldsymbol{w})}{\sum_{b\in\mathcal{A}} \exp(\beta Q^\pi(s,b|\boldsymbol{w}))} \tag{3}$$

where $\beta \in [0,\infty)$ defines the level of rationality.

D. Related Works

Instead of learning from users' demonstrations [17], [18], existing methods rely on querying users for their preferences between trajectories as Eq. 2 [10], [16], [19]. However, these preference queries are not very informative, since they only provide information relative to one other trajectory [9]. To address this issue, active preference learning methods have been proposed, which generate the most informative query at each step [11], [16]. These methods may still require many unnecessary queries when dealing with complex and high-dimensional behavior spaces. Unlike these methods, this paper proposes a new visual analytics approach to ease the labelling effort of the users and avoid a repetitive query-answer process.

III. Visual Analytics for Reward Tuning

This section introduces the design overview of the approach, and then presents the user interface of the visualization system for interactive reward tuning. Also, we describe how to incorporate this system into operations alongside the workflow for preference elicitation. We demonstrate a use case in the Mujoco environment.

A. Design Overview

The design of the visualization system follows the visual-information-seeking principle [13]. To improve the efficiency of the user queries, this system gives users the freedom to dictate the direction of exploration and thus the search direction. The system has two crucial characteristics related to this: 1) it provides an overview of the policies' behaviors as contextual information, and 2) it incorporates visual analytics features to support the user's decision-making process.

The design concept is illustrated in Fig. 1-b. From a user point-of-view, it is an interactive user interface that consists of two interactively linked views: an *Embedding View* and

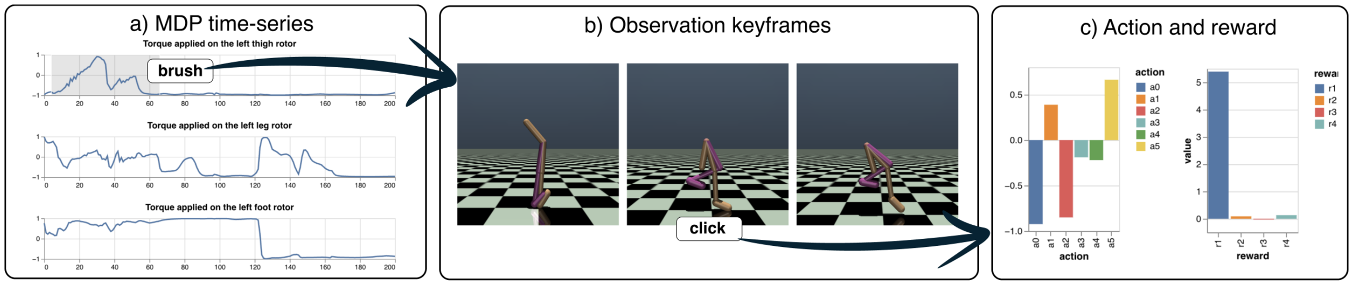

a) MDP time-series
Torque applied on the left thigh rotor
brush
Torque applied on the left leg rotor
Torque applied on the left foot rotor
b) Observation keyframes
click
c) Action and reward
action
reward
value

Fig. 3: When selecting a single behavior, the user can analyze the MDP time-series data in a line chart (a). Brushing on the line chart initiates selecting a range of timesteps and extracting the keyframes (b). Each keyframe comes with bar charts displaying the corresponding actions and rewards (d). These components support the interactions in inspecting and assessing a behavior trajectory.

Sample View. The former displays a contextual overview showing the relationships among behaviors, while the latter provides detailed information based on the user's selection. When a user selects a group of points, the visualization system lets the user gain a better understanding of the behaviors by providing sample videos and visualizations of detailed MDP time-series data. This interactive visual analysis process assumes that we can pre-train a model containing an optimal policy for every possible user preference, considering different objectives [5], [20], [21].

B. *User Interface*

Within the visualization system, users can explore the behavior space, examine groups of sampled behaviors, zoom in on promising areas, and quickly find satisfactory candidates. The approach significantly reduces the number of queries required to find an optimal or satisfactory candidate. Its method assumes that users possess at least basic analysis skills that enable estimating the value of such groups and making decisions at a relevant rational level. Our work demonstrated that the proposed system faithfully follows this assumption. Below, we describe the visualization system's interface (Fig. 2), presenting the design details of each view.

1) Embedding View: The Embedding View is designed to give an overview of all possible behaviors generated from the candidate policies. These possible policies are the results of optimizing different instances of reward terms sampled from the reward distribution. To measure the similarity between behaviors, we apply the dynamic time-warping (DTW) algorithm [22] to the observations over time. The DTW algorithm can compute the temporal alignment between two time-series data and derive a comprehensive distance. From this distance, the behavior data can be captured in a two-dimensional scatterplot (see Fig. 2, a) through a dimensionality-reduction algorithm. Our implementation uses t-SNE [23] for this purpose. Each point plotted represents a specific behavior with corresponding reward weights. Close points, denoting similar behaviors, are more likely to be located in a cluster. Note that while t-SNE with DTW is used here, other methods that work on the data can also be used.

The scatterplot permits the user to perform selection through brushing, thus allowing users to steer the exploration in the behavior space. Upon brushing of a set of points in the scatterplot, the corresponding reward-weight distribution gets visualized for the parallel coordinates, as shown in the figure's pane b. The range for each coordinate is 0 to 1. For example, the selected behaviors in this pane were trained from a reward with a small $w1$ and a large $w4$. $w2$ and $w3$ have a negative correlation relationship. This distribution information can be used to re-train the policy in the outer loop further.

2) Sample View: The Sample View is a feature that provides detailed analysis of selected behaviors. It generates videos sampled from the chosen behaviors as previews to help users understand them better (illustrated in Fig. 2, c). Additionally, it presents all the MDP time-series data for the behaviors as band time-series charts (as pane d shows). A narrow band indicates that the selected behaviors are highly similar and that the sample videos might represent the cluster well. In other cases, further queries within the cluster may be necessary. Using the time-series chart, users can apply their expertise to analyze the behaviors more efficiently without watching all the videos. They can observe trend patterns or specific values for the timesteps and even add their own filters for analysis.

If only one behavior is selected, the band time-series charts appear as line charts (see Fig. 3). For more detailed analysis, users can brush a range of timesteps to extract keyframes on the basis of zero-crossing points [24]. The keyframe snapshot is then shown as a sequence of images. By selecting each keyframe, users can visualize the instant action and reward with bar charts for easy analysis.

C. *Preference Elicitation via Visualization System*

With our visualization system, we elicit user preferences by obtaining feedback to infer the hidden user utility. Algorithm 1 expresses the workflow, which follows three main steps: 1) In a pre-training phase (*lines 1–2*), a meta-policy is trained with random samples from the reward distribution, roughly in the manner of prior pertinent approaches [5], [25], [26]. The trained policy functions as an optimal policy for exploration of the reward space, in that its behavior is conditioned on the reward weights. 2) In preference elicitation (*lines 3–11*), the behavior data are generated from the trained

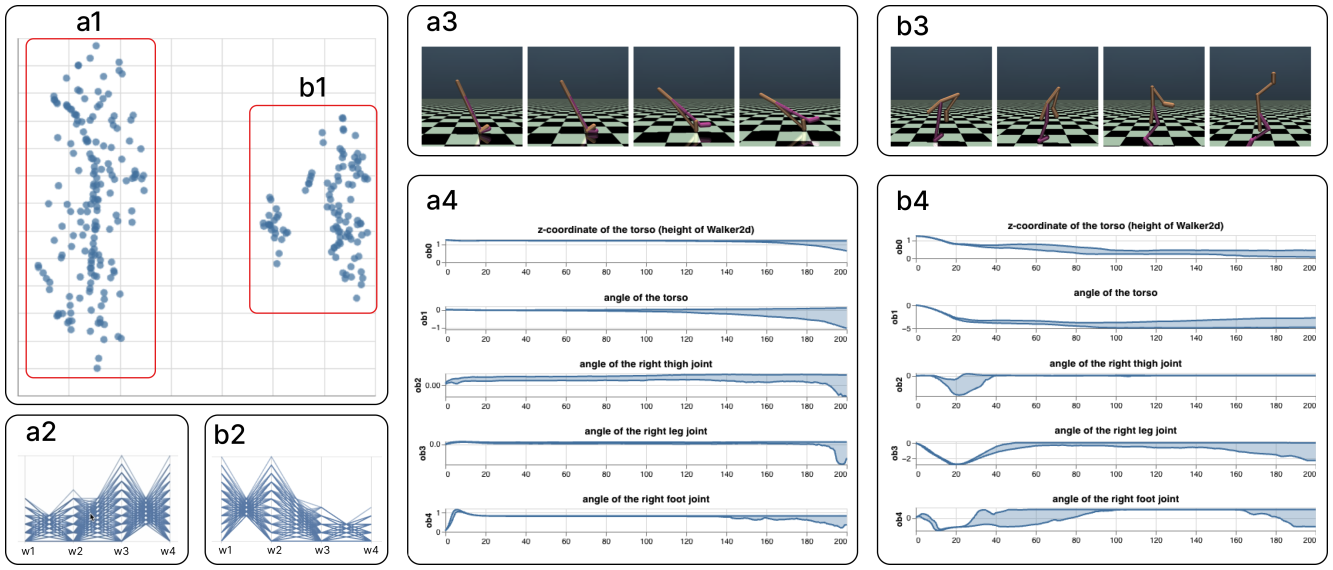


Fig. 4: The visualization system enables interactive visual analytics from a high-level overview to an in-depth analysis. The user begins by examining the larger cluster on the left (a1). The cluster assigns more weight to $w3$ and $w4$ (a2). One sample behavior simply keeps the body straight and falling down (a3). The observation features are in narrow bands over time (a4), reflecting very similar behaviors in the cluster. In the second cluster (b1), greater weight gets assigned to $w1$ and $w2$ (b2). The sample video, presenting attempts to do the splits (b3), indicates that this cluster holds greater promise. Further analysis is required, since the corresponding MDP time-series data show more variety (b4).

policy (*lines 4–7*) and embedded in a low-dimensionality space for visualization (*line 8*), and our system is used to tune the weights through visual analysis (*line 9*). Finally, 3) in a fine-tuning phase (*line 12*), an optimal policy can be found via fine-tuning of the policy once the reward weights have been tuned to the user-preferred combination.

Algorithm 1 Preference Elicitation Workflow

```
Require: The distribution over the reward preference $p(\boldsymbol{w})$
 1: Randomly initialize the meta-policy $\pi_\theta$
 2: Pre-train meta-policy $\pi_\theta$ with $p(\boldsymbol{w})$
 3: while optimal $\boldsymbol{w}^*$ has not been found do
 4:     for each $i$ do
 5:         Sample $\boldsymbol{w}_i \sim p(\boldsymbol{w})$
 6:         Generate trajectory $\tau_i$ following $\pi_\theta(a|s, \boldsymbol{w_i})$
 7:     end for
 8:     Embed $[\tau_0, \ldots, \tau_n]$ based on the trajectory similarity
 9:     Interactively analyze $(\boldsymbol{w}_0, \tau_0), \ldots, (\boldsymbol{w}_n, \tau_n)$
10:     Update $p(\boldsymbol{w})$
11: end while
12: Finetune $\pi_\theta$ to $\pi_\theta^*$ with the optimal $\boldsymbol{w}^*$
```

D. Usage Example

The visualization system can support the environments associated with the Gym API. Let us demonstrate its use in reward tuning by considering an example environment with Walker2D implemented with Gymnasium [1]. The models are trained using the Soft Actor-Critic method [27]. We take the task of *doing the splits* as implemented in this environment. The reward function is defined as the linear combination of reward terms $r = w_1 \cdot r_1 + w_2 \cdot r_2 + w_3 \cdot r_3 + w_4 \cdot r_4$, where r_1 is the angle between the thighs ($r_1 = |obs[5] - obs[2]|$), r_2 is the proximity of the ground ($1 - (obs[0] - 0.4 \cdot sin(obs[1]))$), r_3 is the legs' straightness ($r_3 = -|obs[3]|/2 - |obs[6]|/2$), and r_4 is the straightness of the torso ($r_4 = -|obs[1]|$). The goal is to arrive at the correct weight vector (w_1, w_2, w_3, w_4).

Given the behavior data, the Embedding View displays two distinct clusters of behaviors. The user begins with these, examining the larger cluster (on the left) by brushing certain points and clicking the analysis button (a1), which triggers depicting the reward-weights distribution (a2) as the information to inform analysis for the reward function and the corresponding Sample View. Consulting the Sample View, the user is presented with behavior videos (a3) as the visual information for the policy, along with the time series of MDP features from one episode associated with the behaviors (a4). In this case, the video shows that the behaviors within the cluster considered do not fulfil the task's remit: the body remains straight. These sampled videos are deemed representative of the cluster because nearly all observation features in the MDP series are in narrow bands over time (Fig. 4-a4). The reason for the failures in this cluster might be favoring $w3$ and $w4$ excessively in the weighting (see Fig. 4-a2), whereby these policies put too much effort into maintaining the straightness of the body. In light of this analysis, the user moves on to the second cluster, on the right (in-b1), where the distribution of reward weights shows its favor on $w1$ and $w2$ (b2). The sample videos show attempts to do the split (b3). The MDP time

[1]https://gymnasium.farama.org/environments/mujoco/walker2d

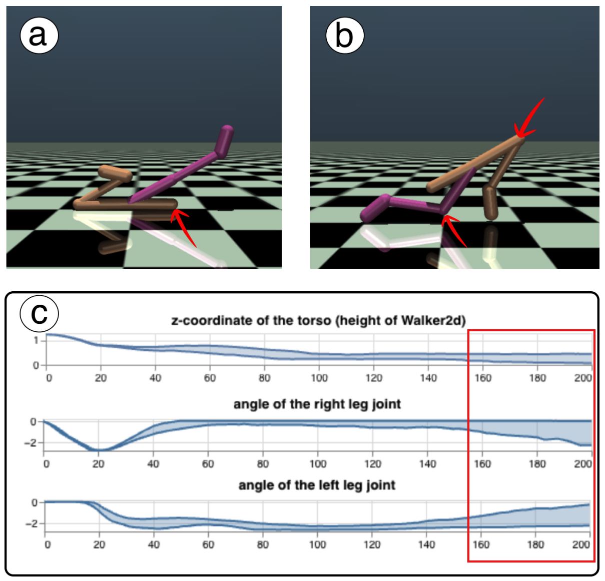


Fig. 5: Two common mistakes have been identified: the agent sometimes falls over at the end, with the torso ending up on the ground (a), and in some cases the agent fails to keep its legs straight (b). These two classes of erroneous behavior can be identified by the z-coordinate of the torso being close to 0 while the angle of the left and right leg joints deviates from 0, as is visible from the patterns in the area marked in red (c). This analysis lets the user filter out undesired behaviors quickly without needing a large number of videos.

series (b4) shows more variety, requiring further analysis.

How visual analytics information aids in decision-making is clear from Fig. 5, depicting the user inspecting the second cluster in the Embedding View. Here, the videos show better attempts at the splits, though small mistakes in the behavior still lead to failure. The user identifies a common mistake evident in the videos: the agent tips over, and its torso ends up on the ground (see Fig. 5-a). When the agent falls, the $ob0$ value (the z-coordinate of the torso) approaches 0 (as the plots in pane c show), because the torso is now at ground level. Therefore, to eliminate this unsatisfactory behavior from consideration without tedious comparison of videos, the user could filter out behaviors with $ob0$ values close to 0 at the end of the trajectory. The user continues to explore the Embedding View. For example, doing the splits properly involves keeping both legs straight at the end (see Fig. 5-b). Accordingly, $ob3$ (the relevant angle for the right leg) and $ob6$ (the corresponding one for the left) should be close to 0 when the trajectories end, so the user includes these two conditions for tuning the weights.

After these queries, the user narrows the search to a small space and inspects the behavior at each of the final few points individually. After considering each video and analyzing the MDP series data, the user can determine the optimal parameters. The final tuned reward, $r = 0.2 \cdot r_1 + 0.4 \cdot r_2 + 0.3 \cdot r_3 + 0.2 \cdot r_4$, can successfully align the agent to do the splits. With the reward, the model can be further fine-tuned.

IV. Simulation-based Evaluation

This section presents results from a simulation study. It shows the effectiveness of analyzing groups of sampled behaviors through an embedded view, demonstrating that it requires less user effort. In general, we describe the environment setup, then introduce the simulated users, and finally look at the analysis results.

A. Environment Setup

In the setup we employed, we prepare the behavior datasets by randomly generating 200 points in the embedding space. For simplicity, we assume the embedding space is the same as the embedding view in our system, which is two-dimensional. Each point c_i represents a behavior with a feature vector $\phi(c_i)$ based on the x–y coordinates, since we are assuming that the relationship of the behaviors is captured well through the embedding approach. The ground-truth candidate (denoted as c_t) is chosen randomly from among the points. We use utility measurement to evaluate performance on the basis of the similarity between the ground-truth candidate and the inferred candidate (denoted as c_j) within a given space. This value is defined as $1 - ||(\phi(c_i), \phi(c_t)||_2$, where the latter term represents the normalized distance between the two candidates. In this environment setting, the goal is to find the ground-truth candidate through queries, and we find the number of queries needed to identify it.

B. Simulated Users

We built simulated users by modeling how they perform queries and how they show their preferences.

1) User Query: We defined four types of user queries – three using baseline techniques and our approach:

- *Pairwise comparisons* [28]: The user chooses a winner from between two items, which is compared to a new item with the next query.
- *Ranking* [28]: The user starts by ranking two items; then, each subsequent round adds one more item, which has to be sorted into the rank order. The information is of higher quality with the ranking approach as compared to pairwise comparison when the quantity of queries is the same.
- *Clustering* [28]: Instead of performing a full ranking, the user selects the best item and groups the remaining ones into clusters. Items with similar utility values are placed in the same cluster.
- *Our approach*: User queries with our approach proceed in a top-down hierarchical fashion from high-level to low-level clusters (see Subsec. III-D). For simplicity, we assume that each high-level cluster contains two sub-clusters (in practice, one can use an arbitrary number of sub-clusters). Given two clusters, the user measures the average utility in each and selects a winner cluster for further inspection. The process iterates until reaching the final behavior remaining.

2) User Preference: We modeled the user's preferences via the noisy model and Boltzmann rational model [15].

- *Noisy model*: We measured the performance of the queries under the noisy-user model defined in Eq. 2. It uses Gaussian processes (GPs) for preference elicitation [28].
- *Boltzmann rational model*: By assuming that the choice of groups can influence the human designer's capability level and presuming that users are Boltzmann rational, we defined the user model as a Boltzmann rational agent (see Eq. 3) that reflects selection of the best cluster c_i alongside the other clusters in queries:

$$p(c_i|\boldsymbol{w}, \delta) = \frac{\exp(\beta\hat{R}(c_i|\boldsymbol{w}))}{\sum_j^k \exp(\beta\hat{R}(c_j|\boldsymbol{w}))} \tag{4}$$

C. Results

For each setting, we used 50 iterations with 50 queries. Each iteration has a dataset with 200 randomly generated data points. Our approach outperforms all baselines with both user models. For the noisy model, we can see from Figure 6 that our method is robust to varying levels of noise – the user model considers the average performance of the clusters since all feedback consists of noisy estimates up to a constant noise level. Our approach uses, on average, 15 queries to reach an average utility of 0.954, which is higher than that of pairwise comparison (0.929), clustering (0.934), and ranking (0.932). For the Boltzmann rational model, we measured the mean utility and standard error for queries under several levels of rationality. The results (in Fig. 7) attest that our method reaches greater utility with fewer queries when compared to other query types. As is visible from the graphs, our method converges more quickly than other queries; i.e., it prevents unnecessary evaluation of most of the data points available. Our approach uses 15 queries, on average, to reach an average utility of 0.972, better than pairwise comparison's 0.904, clustering's 0.919, and ranking's 0.914.

V. Conclusion and Discussion

The novel visual analytics approach presented here enables users to interactively search for desired reward parameters. Our system allows users to control their insight and integrate it into the preference-elicitation process. The simulation study proves that our approach can outperform baseline methods, requiring less user effort. The proposed system demonstrates several advantages:

A1 It offers a more intuitive and efficient way to support users' decision-making process with a structured Embedding View accompanied by analytics data.

A2 The system allows users to gain a deeper understanding of the reward landscape and the link between the policies' behavior patterns and their associated reward parameters.

A3 This lets users choose and inspect policies with complex behaviors freely, in accordance with their preferences.

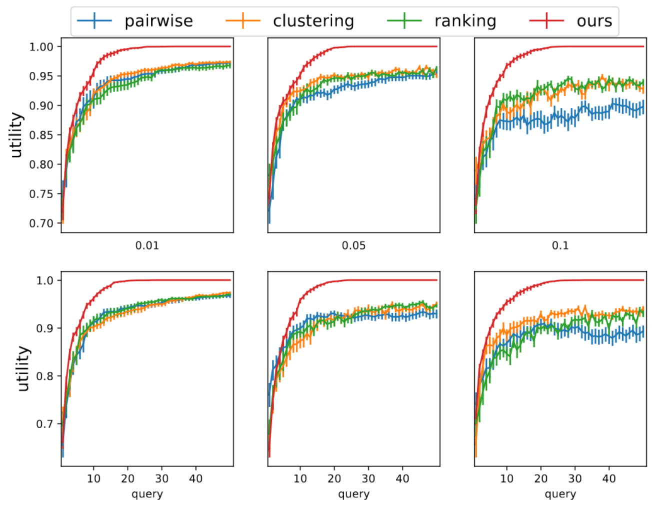


Fig. 6: Our approach outperforms baseline ones under the noisy model with three noise levels: 0.01, 0.05, and 0.1. The first row shows the GP without a prior, and the second shows the one with a GP that has a linear prior.

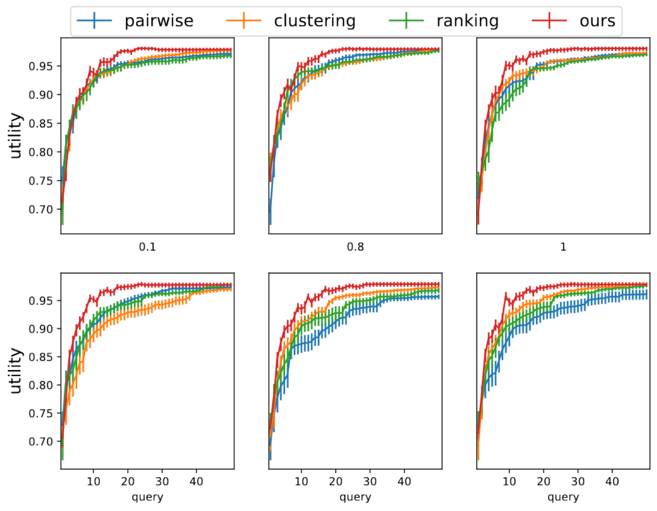


Fig. 7: Our approach outperforms the baselines under the Boltzmann rational model with three rationality levels: 0.1, 0.8, and 1. The two rows show the GP without a prior and with a linear prior.

However, it also necessitates tradeoffs and shows limitations relative to standard pairwise comparisons: 1) Our approach assumes a user with some expertise in visual analysis. For those who lack this skill, supplying a tutorial and training is required, which might end up more time-consuming than the standard pairwise comparison methods. 2) The simulated physical environment in RL is limited to visual rendering, which may not be applicable to all problems. 3) Currently, the system handles only reward functions that are limited to linear expressions. This limits work with more complex cases in real-world problems. Also, future efforts could explore extending the visualization system to support addressing problems in reinforcement-based learning from human feedback without any reward functions [29].

ACKNOWLEDGMENT

This work was supported by the Research Council of Finland (flagship program: Finnish Center for Artificial Intelligence, FCAI, grants 328400, 345604, 341763; Human Automata, grant 328813;Subjective Functions, grant 357578).

REFERENCES

[1] S. Singh, R. L. Lewis, and A. G. Barto, "Where do rewards come from?" in *Proceedings of the Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 2009, pp. 2601–2606.

[2] S. Mahadevan, "Average reward reinforcement learning: Foundations, algorithms, and empirical results," *Machine Learning*, vol. 22, no. 1, pp. 159–195, 1996.

[3] W. B. Knox and P. Stone, "Learning non-myopically from human-generated reward," in *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, 2013, pp. 191–202.

[4] D. Gupta, Y. Chandak, S. Jordan, P. S. Thomas, and B. C da Silva, "Behavior alignment via reward function optimization," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[5] X. Chen, A. Ghadirzadeh, M. Björkman, and P. Jensfelt, "Meta-learning for multi-objective reinforcement learning," in *Proceedings 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 977–983.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[7] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis)design for autonomous driving," *Artificial Intelligence*, vol. 316, 2023.

[8] D. Sadigh, A. Dragan, S. Sastry, and S. Seshia, *Active preference-based learning of reward functions*, 2017.

[9] M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions by integrating human demonstrations and preferences," *arXiv preprint arXiv:1906.08928*, 2019.

[10] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, vol. 18, no. 136, 2017.

[11] A. Wilson, A. Fern, and P. Tadepalli, "A bayesian approach for policy learning from trajectory preference queries," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1142–1150, 2012.

[12] Y. Koyama, I. Sato, and M. Goto, "Sequential gallery for interactive visual design optimization," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, 2020.

[13] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*. IEEE, 1996, pp. 336–343.

[14] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," in *ICML '10: Proceedings of the 27th International Conference on Machine Learning*. ACM, 2010, pp. 1255–1262.

[15] C. Laidlaw and A. Dragan, "The Boltzmann policy distribution: Accounting for systematic suboptimality in human models," in *International Conference on Learning Representations*, 2021.

[16] R. Akrour, M. Schoenauer, and M. Sebag, "APRIL: Active preference-learning based reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 116–131.

[17] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning." in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[18] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML '04: Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[19] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park, "Preference-based reinforcement learning: A formal framework and a policy iteration algorithm," *Machine Learning*, vol. 89, no. 1–2, pp. 123–156, 2012.

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

[21] D. J. Lizotte, M. Bowling, and S. A. Murphy, "Linear fitted-Q iteration with multiple reward functions," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3253–3295, 2012.

[22] S. Salvador and P. Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," in *KDD Workshop on Mining Temporal and Sequential Data*, vol. 6. Seattle, Washington, 2004, pp. 70–80.

[23] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE." *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[24] Y. Yang, L. Zeng, and H. Leung, "Keyframe extraction from motion capture data for visualization," in *2016 International Conference on Virtual Reality and Visualization (ICVRV)*. IEEE, 2016, pp. 154–157.

[25] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," *arXiv preprint arXiv:1907.01657*, 2019.

[26] A. Kumar, A. Singh, F. Ebert, Y. Yang, C. Finn, and S. Levine, "Pre-training for robots: Offline RL enables learning new tasks from a handful of trials," *arXiv preprint arXiv:2210.05178*, 2022.

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[28] L. M. Zintgraf, D. M. Roijers, S. Linders, C. M. Jonker, and A. Nowé, "Ordered preference elicitation strategies for supporting multi-objective decision making," *arXiv preprint arXiv:1802.07606*, 2018.

[29] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1133–1141, 2017.