Eurographics Conference on Visualization (EuroVis) 2025
W. Aigner, N. Andrienko, and B. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 3

Interactive Groupwise Comparison for Reinforcement Learning from Human Feedback (Supplemental Material)

Anon. Submission Id: 1146

In this supplemental material, we present a comprehensive analysis of the design choices (Sec. 1), explain the tree structure resulting from the hierarchical clustering (Sec. 2, explain how the 3 cases of our case study in the main paper were achieved (Sec. 3, and finally illustrate the environments utilized in the synthetic study (Sec. 4).

1. An Analysis of Design Choices

The trajectory data clusters were generated using dynamic time warping. To evaluate the effectiveness of the information visualization for our objectives, we calculated the intra-cluster variance (refer to Figure 1) of the rewards based on the true reward function. Paired t-tests revealed a statistically significant difference between hierarchical clustering and both PCA and t-SNE; however, there was no statistically significant difference between PCA and t-SNE.

To be exact, we generated 50 trajectories for n=10 times and used the three different methods (PCA, t-SNE, and hierarchical clustering) with DTW to dimensionally reduce them. The task that the user ultimately solves with the visualization is to identify two clusters and to compare them. Therefore, they need to select clusters of behaviors in each of the views. In the scatter plots it can be done by lassoing close points. In the hierarchical plot it can be done by selecting tree nodes. In our analysis, we clustered the dimensionally reduced data:

- in the case of PCA and t-SNE we clustered the data points with DBSCAN and ignored all the outliers
- and in the Hierarchical Clustering by selecting the nodes of level 2 that have more than 1 descending leaf node (outliers)

This produces on average 10 clusters in each of those methods. Next, we computed the intra-cluster variance in the true reward of the found clusters and performed a t-test on those variances.

The results are the following:

1. Paired t-test results (t-SNE vs PCA): t-statistic = -0.4588, p-value = 0.6573
   There is no statistically significant difference between t-SNE and PCA.
2. Paired t-test results (t-SNE vs Hierarchical): t-statistic = 4.4657, p-value = 0.0016
   There is a statistically significant difference between t-SNE and Hierarchical. Hierarchical is better than t-SNE.
3. Paired t-test results (PCA vs Hierarchical): t-statistic = 6.5426, p-value = 0.0001
   There is a statistically significant difference between PCA and Hierarchical. Hierarchical is better than PCA.

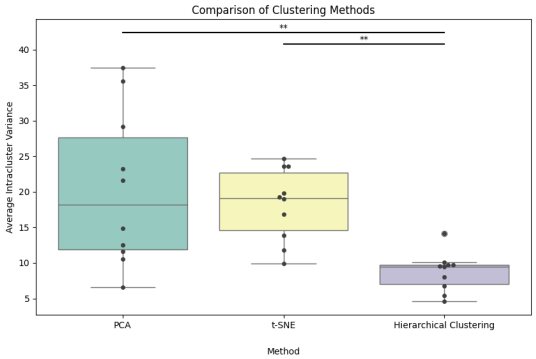


Figure 1: A comparison among PCA, t-SNE, and hierarchical clustering shows that hierarchical clustering exhibits significantly lower variance than the others.

2. Hierarchical Clustering - Dendrogram node order

Even though it might seem as the horizontal order of the tree nodes in the dendrogram showing the n-ary tree structure is arbitrary, it actually stems from the agglomerative clustering process. The agglomerative clustering gives a binary tree structure. The binary tree is converted to an n-ary tree where the horizontal order of the nodes is well defined. The conversion to an n-ary tree structure hides some details.

Lets say there are children $c1$ with descending leaf nodes a, b, c, and d and $c2$ with descending leaf nodes e and f. They both decend from the parent $p1$. $p1$ has a sibling whose child is the leaf node i. In the conversion from the n-ary tree to binary tree, all parent nodes of a level except the topmost node are pruned, i.e. only the relationsips that cross the threshold are kept. So if $p1$ is pruned, we can display the n-ary tree structure in many different ways. For example, we could show the nodes a, b, c, d, e, and f in the same

© 2025 The Author(s). Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

order as before, or we could insert the leaf-node i between d and e. However, this would violate the binary tree structure that the agglomerative clustering resulted in. See figure 2 for an illustration.

The nodes can be reordered horizontally also in the binary tree, but we can only switch the places of two child nodes. They always have to be together (no other nodes can be inserted between them) or the relationships would cross each other's paths. In the example of the figure 2, where i has the same parent as the parent of nodes g and h, we could have the orders $g-h-i$, $h-g-i$, $i-g-h$, and $i-h-g$ but never $g-i-h$ or $h-i-g$: the sibling relationship of g and h must not be violated.

Therefore, it might look like the nodes could be reordered arbitrarily. However, when displaying the n-ary tree structure, we should honor the parental relationships of the pruned nodes from the binary tree as to not loose the order that the agglomerative clustering resulted in. This is easy, we can just keep the order of the leaf nodes that the binary tree showed.

3. Example Cases

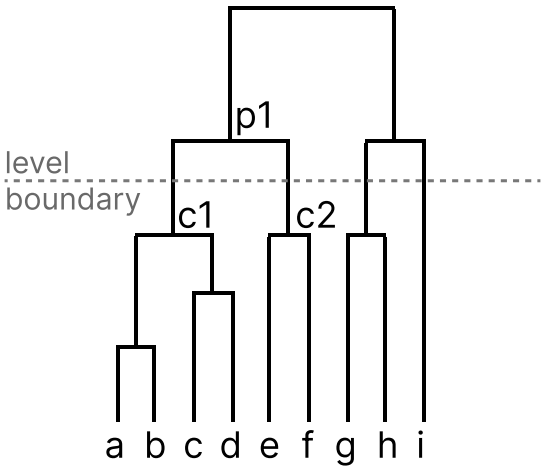
In this section, we demonstrate our approach in 3 MuJoCo environments [TET12] (HalfCheetah, Walker, and Hopper) that are commonly utilized in research on RL and RLHF [CLB*17]. Mujoco is a popular free and open-source physics engine that aims to facilitate research and development in Robotics, Biomechanics, Graphics and Animation, and other areas where fast and accurate simulation is needed [TET12]. Our objective is to train policies to execute behaviors for which there are no predefined reward functions - therefore, RLHF can help us to train such policies. Using our interface, we could deliberately provide comparisons that are related to the final intended behavior.

To teach the agent a specific behavior, the expert user begins by utilizing the Exploration View. The user can click on segments of the radial chart to examine different groups of behaviors, allowing for both pairwise and groupwise comparisons. After selecting two groups, the user can observe and compare the behavior videos in the Comparison View. The expert then chooses the preferred group that aligns better with the intended objective. Once the comparative feedback is provided, the interface records this preference and draws a line of preference in the chart. This feature helps users recall the preferences they expressed before.

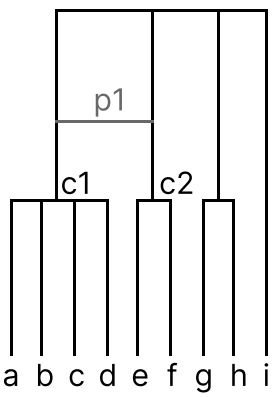
The expert user chooses to explore the behavior space by identifying regions that have not yet been analyzed based on the comparison history, and other times by following suggestions based on the density of linked lines in the Exploration View. Using the Exploration View, the user not only identifies unexplored regions, but also ensures that there is a good spread of comparisons (most behaviors are included in a comparison and connections span in different directions) and gives their favorite behaviors more wins over different competition, and their least favorite more losses. This alternating between exploration and comparison continues until the reward model is well trained.

3.1. Trained Behaviors

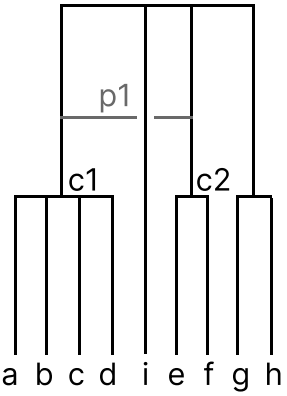
Three policies have been trained to perform novel behaviors without ground-truth rewards (see Fig. 3). While the quality of behaviors in relation to the goal behavior are simple to recognize, defining a mathematical reward function that outputs the reward from the input of observations and actions is not straightforward.



(a) The binary tree structure. The level boundary is used to convert the binary tree into an n-ary tree by pruning all intermediate nodes that come before the threshold, i.e. only keeping the relationships that cross it.



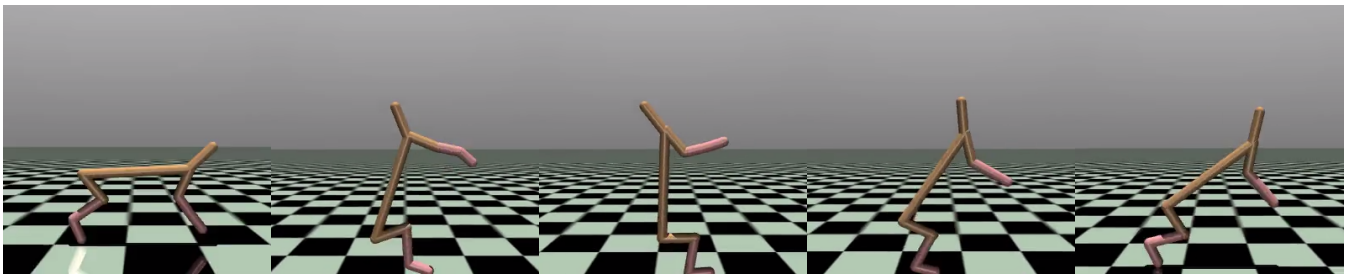
(b) A valid display of the n-ary tree. $c1$ and $c2$ are next to each other. Their relationship coming from their pruned parent $p1$ is not violated.



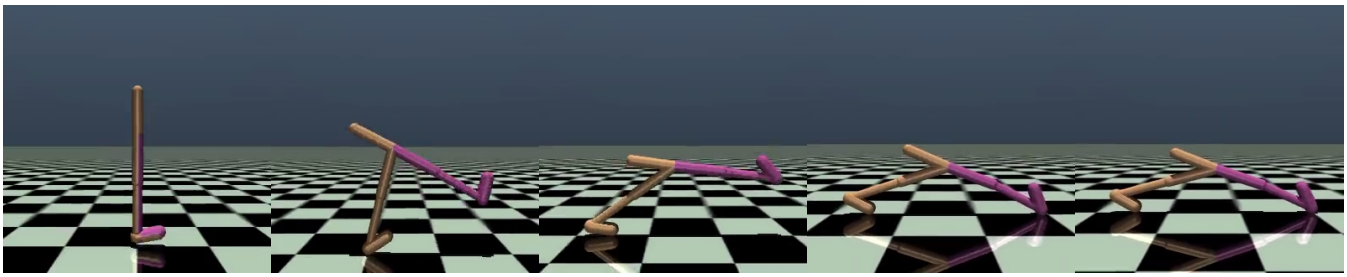
(c) An invalid display of the n-ary tree. $c1$ and $c2$ are divided by the leaf node i. Their relationship coming from their pruned parent $p1$ is violated.

Figure 2: The binary tree structure is converted to an n-tree by splitting it at certain heights and pruning nodes. If we display the n-ary tree, we should keep the order of leaf nodes that we had in the binary tree as to not lose the information of the agglomerative clustering.

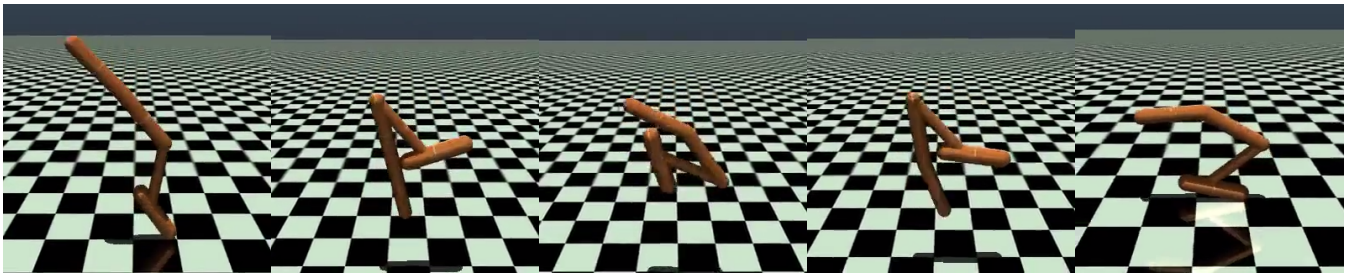
© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.



(a) HalfCheetah learned to stand up from 601 preferences.



(b) Walker learned to do the splits from 616 preferences.



(c) Hopper learned to double backflip from 1032 preferences.

Figure 3: Behaviors that do not have ground-truth rewards can be effectively learned from interactive groupwise comparisons. We show 5 sequential frames for each behavior, and the video can be found linked in our project page.

4. Simulation Environments

The simulation study was conducted in the following six RL environments: Hopper, Walker, HalfCheetah, Reacher, GridWorld, and ContinuousMountainCar. Figures 4, 5, 6 show examples of three environments in our user interface.

All the environments were changed in a way that the episodes were all of uniform length. Because premature episode termination would result in a signal to the agent of the desired behavior. For example, Christiano et al. [CLB*17] use fixed horizon environments because: Removing variable length episodes leaves the agent with only the information encoded in the environment itself; human feedback provides its only guidance about what it ought to do.

The reward functions of the Hopper, Walker, HalfCheetah, and Reacher could be used as is for the simulation study. The reward function of the ContinuousMountainCar and the Gridworld had to be slightly adapted to change the reward from a sparse reward only given at successful termination to a dense reward that facilitates the comparison of intermediate steps and a fixed episode length. Crucially, the goal of those two environments remained the same, the agent just needed to remain in that goal position after reaching it to earn more reward until reaching the fixed end of the episode.

References

[CLB*17] Christiano P. F., Leike J., Brown T. B., Martic M., Legg S., Amodei D.: Deep reinforcement learning from human preferences. *ArXiv abs/1706.03741* (2017). URL: https://api.semanticscholar.org/CorpusID:4787508. 2, 3

[TET12] Todorov E., Erez T., Tassa Y.: Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (2012), IEEE, pp. 5026–5033. 2

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

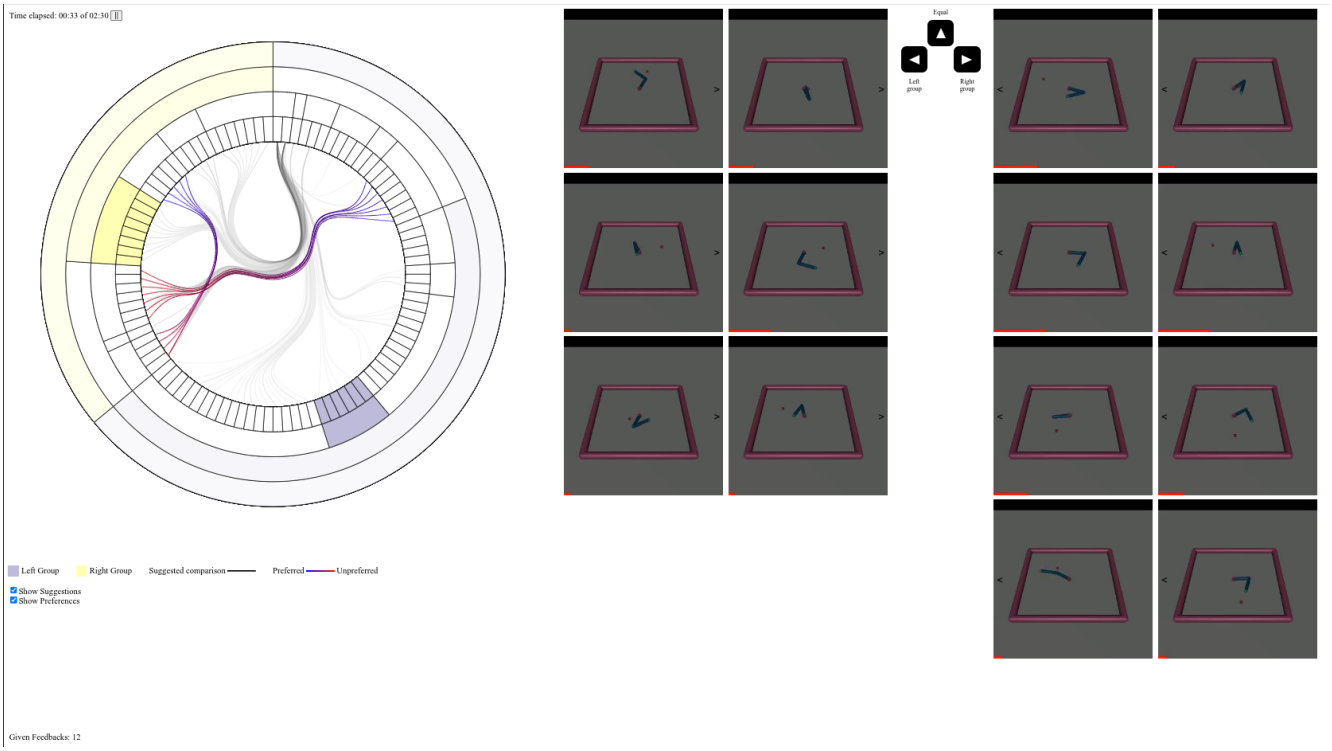


Figure 4: Groupwise Comparison of behaviors of the Reacher environment in our user interface

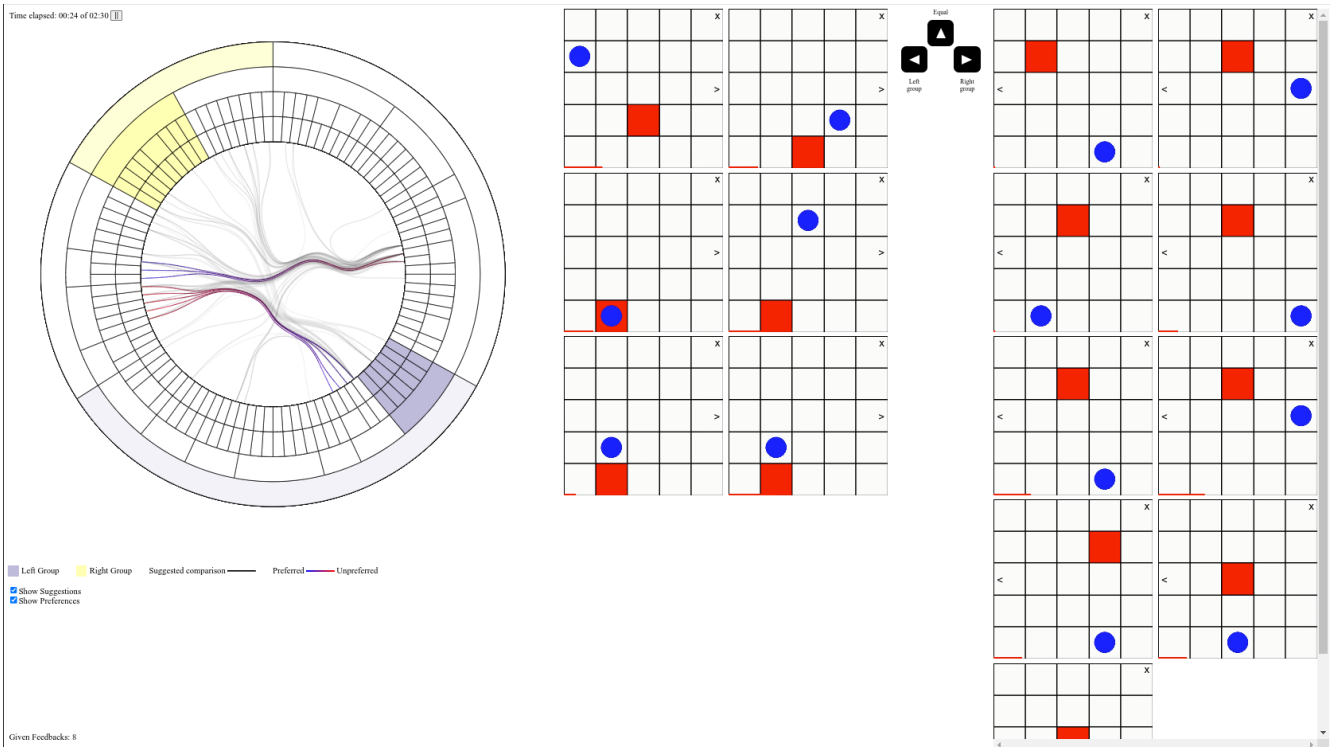


Figure 5: Groupwise Comparison of behaviors of the GridWorld environment in our user interface

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

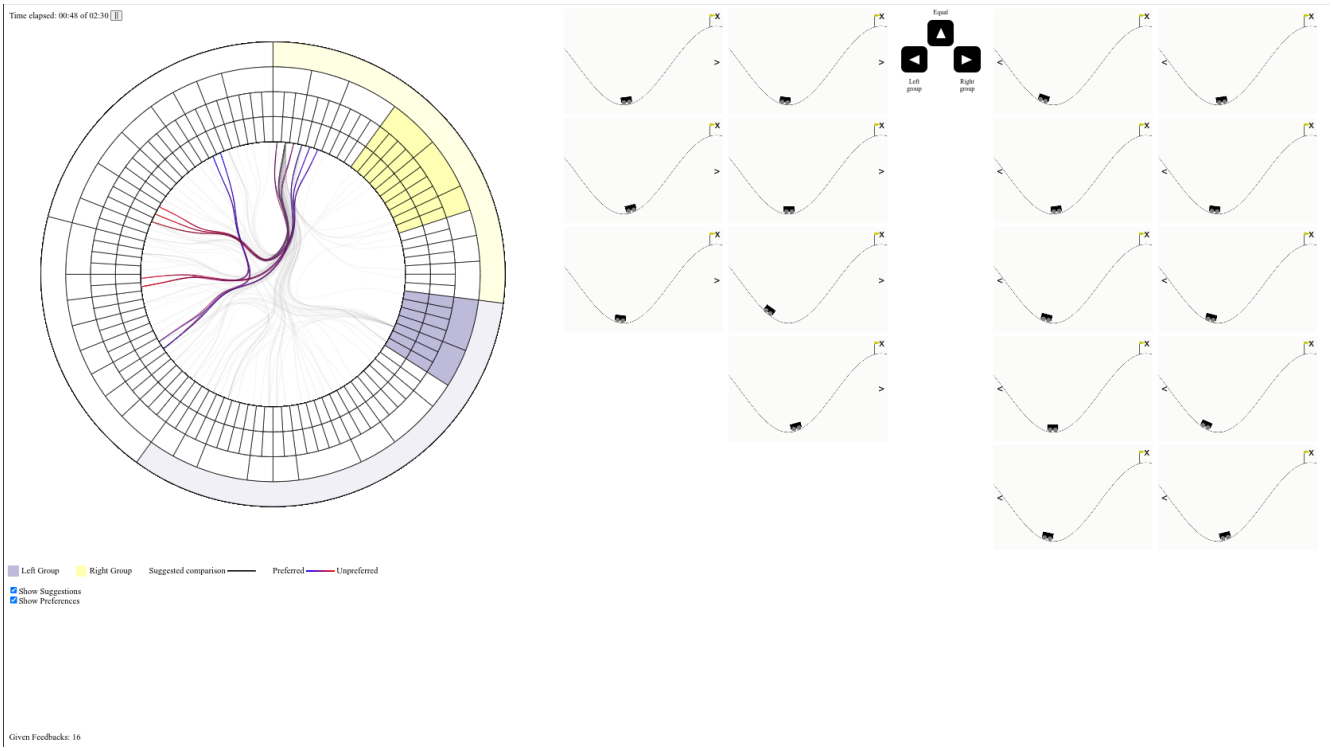


Figure 6: Groupwise Comparison of behaviors of the ContinuousMountainCar environment in our user interface

© 2025 The Author(s).
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.