Supplementary Material for "Sequential Preference Ranking for Efficient Reinforcement Learning from Human Feedback"

We provide additional proofs, algorithm details, experiment details, and analyses of the proposed method in this supplementary material. Section A summarizes the definitions of symbols and terms used in the paper and the supplementary material. Section B provides proofs for Lemma 1 to Lemma 4, Theorem 1 to Theorem 3, and Corollary 3.1. Section C describes two main algorithms for RLHF using sequential and root pairwise comparison. Section D provides detailed experiment settings and detailed analyses.

A Summary of Notations

The definitions of symbols and terms described in Greek letters or alphabets are summarized in Table 1. The alphabets in the table are arranged in alphabetical order.

Symbol / Term	Definition
σ	fixed-length trajectory segment
σ_i	i^{th} sampled trajectory segment
$\sigma_i \succ \sigma_j$	σ_i is preferred to σ_j
$\sigma_i \prec \sigma_j$	σ_i is preferred to σ_j
θ	estimated reward model parameter
θ^*	true reward model parameter
ϕ	trajectory encoder
ψ	1-dim discrete action for real robot experiment
$\Delta(G)$	the maximum degree of a dependency graph G
$\Delta_{M,t}$	the maximum degree of a dependency graph G_t generated at the t-th iteration by comparing M segments per iteration
α	coefficient of M in the linear approximation function of a_M
β	$\mathbb{E}[S_{M,T}]$ divided by TM
γ	intercept of the linear approximation function of a_M
η	feedback efficiency
ν_ϕ	learning rate of policy update
ν_θ	learning rate of reward update
1K steps	$1,000$ steps of training the agent policy

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

a_M	$\mathbb{E}[S_M]$, the number of trajectory segment pairs by comparing M trajectory segments using sequential pairwise comparison
B	$\ln\left(1+\exp\left(\frac{2S^2M(M-1)}{\kappa\lambda_{\min}(\Sigma_X)}+2SD\right)\right)$
b_M	$\mathbb{E}[S_M]$ for root pairwise comparison
D	upper bound of the norm of $\theta^\star$, i.e., $\\|\theta^\star\\|_2 \leq D$
F	$\left(\frac{2SM(M-1)}{\kappa D\lambda_{\min}(\Sigma_X)}+2\right)S$
$G=(V,E)$	dependency graph G defined with the set of nodes V and the set of edges E
H	time horizon of every segment
$\mathcal{K}$	the set of previously chosen trajectories sampled from $\mathcal{U}$ for reward update
$\mathcal{K}_i$	$\{\sigma_1, \sigma_2, ..., \sigma_i\}$
L_{pref}	cross-entropy loss for predicting the preference label
M	the number of sampled trajectories at each iteration
N	the number of human feedback at each iteration
p_N	the expected total number of segment pairs collected from N human feedback
Q	upper bound of the norm of $\nabla\theta$, i.e., $\\|\nabla\theta\\|_2 \leq Q$
R	true risk of the reward model
$\hat{R}$	empirical risk of the reward model, same with L_{pref}
$r(s,a)$	true reward for a state-action pair (s,a)
$\hat{r}(s,a)$	estimated reward for a state-action pair (s,a)
$\hat{r}_\theta$	a linear reward model parameterized by θ
S	upper bound of the norm of $\phi(\sigma)$, i.e., $\\|\phi(\sigma)\\|_2 \leq S$
S_M	the total number of segment pairs collected among M trajectory segments
$S_{M,T}$	the cumulative sum of the number of training data of the reward model from iteration 1 to T, with M segments compared per iteration
T	number of training iterations of the global iterative process
U	number of training iterations of policy update
$\mathcal{U}$	the set of trajectories in the replay buffer
W	number of training iterations of reward update
X_M	$S_M \mid \sigma_1 \prec \sigma_2$
x	$\phi(\sigma_1)-\phi(\sigma_2)$
Y_M	$S_M \mid \sigma_1 \succ \sigma_2$
y	true preference label
$\hat{y}$	estimated preference label

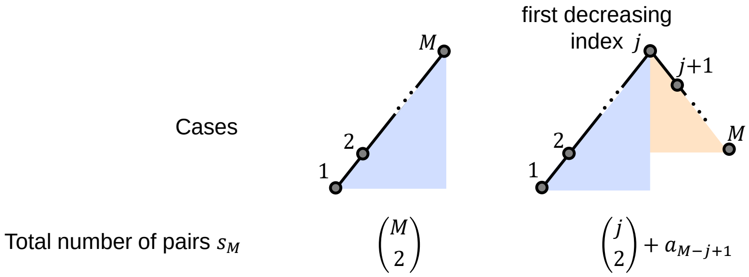
Table 1: **Summary of Notations.**

B Proofs

B.1 Feedback Efficiency of the Algorithm

In this section, we provide detailed proofs for Lemma 1, Theorem 1, and Theorem 2.

B.1.1 Sequential Pairwise Comparison



first decreasing index j
Cases
M
2
1
$j+1$
M
2
1
Total number of pairs s_M
$\binom{M}{2}$
$\binom{j}{2} + a_{M-j+1}$

Figure 1: **Total number of pairs from two cases in sequential pairwise comparison.** The left figure shows an increasing sequence of segments. The right figure shows a non-increasing sequence with the first decreasing index j. Numbers indicate the total number of pairs from all the cases in each scenario. Due to symmetry, we only illustrate sequences starting with an increasing pair.

***Definition* B.1.** *When comparing two segments σ_i and σ_j, we define the pair (σ_i, σ_j) as an increasing pair if $\sigma_i \prec \sigma_j$. If $\sigma_i \succ \sigma_j$, the pair is called a decreasing pair.*

We define a conditional random variable $X_M := S_M|\sigma_1 \prec \sigma_2$, starting the sequence with an increasing pair, and define $Y_M := S_M|\sigma_1 \succ \sigma_2$, starting the sequence with a decreasing pair. By definitions, $S_M = X_M\mathbb{I}[\sigma_1 \prec \sigma_2] + Y_M\mathbb{I}[\sigma_1 \succ \sigma_2]$ and $\mathbb{E}[X_M] = \mathbb{E}[Y_M]$.

***Proof of Lemma 1*.** For the proof of Lemma 1, let us define a random index J that indicates the index of the first appearance of a decreasing pair in a sequence. Note that J occurs within $[2, M-1]$, $\sigma_J \succ \sigma_{J+1}$ holds, and let $J = M$ indicate a decreasing pair not appearing in a sequence. Then, let us derive the probability $\mathbb{P}(J \geq j|\sigma_1 \prec \sigma_2)$ as follows:

$$\mathbb{P}(J \geq j|\sigma_1 \prec \sigma_2) = \frac{\mathbb{P}(J \geq j, \sigma_1 \prec \sigma_2)}{\mathbb{P}(\sigma_1 \prec \sigma_2)} \tag{1}$$

$$= \frac{\frac{\binom{M}{j}(M-j)!}{M!}}{\frac{\binom{M}{2}(M-2)!}{M!}} = \frac{1/j!}{1/2} = \frac{2}{j!}. \tag{2}$$

From $\mathbb{P}(J \geq j|\sigma_1 \prec \sigma_2)$, we have,

$$\mathbb{P}(J = j|\sigma_1 \prec \sigma_2) = \mathbb{P}(J \geq j|\sigma_1 \prec \sigma_2) \tag{3}$$

$$- \mathbb{P}(J \geq j+1|\sigma_1 \prec \sigma_2) \tag{4}$$

$$= \frac{2}{j!} - \frac{2}{(j+1)!} = \frac{2j}{(j+1)!}. \tag{5}$$

Then, we can derive the recurrence relation of a_M as follows:

$$a_M = \mathbb{E}[S_M] = \mathbb{E}[S_M|\sigma_1 \prec \sigma_2]\mathbb{P}(\sigma_1 \prec \sigma_2) + \mathbb{E}[S_M|\sigma_1 \succ \sigma_2]\mathbb{P}(\sigma_1 \succ \sigma_2) \tag{6}$$

$$= \frac{1}{2}\mathbb{E}[X_M] + \frac{1}{2}\mathbb{E}[Y_M] = \mathbb{E}[X_M] \ \ (\because \mathbb{E}[X_M] = \mathbb{E}[Y_M]) \tag{7}$$

$$= \sum_{j=2}^{M} \mathbb{E}[S_M|\sigma_1 \prec \sigma_2, J = j]\mathbb{P}(J = j|\sigma_1 \prec \sigma_2) \tag{8}$$

$$= \binom{M}{2}\frac{2}{M!} + \sum_{j=2}^{M-1} \mathbb{E}[S_M|\sigma_1 \prec \sigma_2, J = j]\mathbb{P}(J = j|\sigma_1 \prec \sigma_2) \tag{9}$$

$$= \frac{1}{(M-2)!} + \sum_{j=2}^{M-1} \mathbb{E}[\binom{j}{2} + Y_{M-j+1}|\sigma_1 \prec \sigma_2, J = j]\frac{2j}{(j+1)!} \tag{10}$$

$$= \frac{1}{(M-2)!} + \sum_{j=2}^{M-1} \frac{2j\binom{j}{2}}{(j+1)!} + \sum_{j=2}^{M-1} \mathbb{E}[Y_{M-j+1}|\sigma_1 \prec \sigma_2, J = j]\frac{2j}{(j+1)!} \tag{11}$$

$$= 2 - \frac{2}{M!} + \sum_{j=2}^{M-1} a_{M-j+1} \frac{2j}{(j+1)!}. \tag{12}$$

$\square$

Proof of Theorem 1. For the base case, let $M = 2$. Then, we have,

$$\left| \frac{a_2}{2} - \frac{1}{e-2} \right| = \left| \frac{1}{2} - \frac{1}{e-2} \right| = \frac{4-e}{2(e-2)} = constant. \tag{13}$$

Hence, the statement holds for $M = 2$.

Now, let us assume that the statement holds for all $n \in [3, M-1]$.

Suppose that there exists a constant $\alpha > 0$ such that $\alpha(n-\gamma) + f(n) = a_n$, where $f(n)$ indicates a remainder function bounded as $|f(n)| < c$ for all $n \in [2, M-1]$. c indicates a finite positive constant. Then, from Lemma 1 and the following Taylor series expansion and its variants

$$\sum_{j=0}^{\infty} \frac{x^j}{j!} = e^x, \sum_{j=2}^{\infty} \frac{j}{(j{+}1)!} = \frac{1}{2}, \sum_{j=2}^{\infty} \frac{j^2}{(j{+}1)!} = e - \frac{3}{2}, \tag{14}$$

we have the following conjecture.

$$|f(M)| \tag{15}$$

$$= \left| -\alpha(M-\gamma) + 2 - \frac{2}{M!} + \sum_{j=2}^{M-1} \left(\alpha(M-j+1-\gamma) + f(M-j+1)\right) \frac{2j}{(j+1)!} \right| \tag{16}$$

$$\leq \left| -\alpha(M-\gamma) + 2 - \frac{2}{M!} + \alpha(M+1-\gamma)\left(1 - \frac{2}{M!}\right) - \alpha \sum_{j=2}^{M-1} \frac{2j^2}{(j+1)!} \right| + c\left|1 - \frac{2}{M!}\right| \tag{17}$$

$$= \left| \alpha + 2 - \frac{2}{M!}(1 + \alpha(M{+}1{-}\gamma)) - \alpha(2e{-}3) - 2\alpha\left(\left(\sum_{j=2}^{M-1} \frac{j^2}{(j+1)!}\right) - e + \frac{3}{2}\right) \right| + c\left|1 - \frac{2}{M!}\right| \tag{18}$$

$$\leq 2 + c - \alpha(2e-4) + \left| \frac{2}{M!}\left(1 + \alpha(M+1-\gamma)\right) \right| + 2\alpha\left| \left(\sum_{j=2}^{M-1} \frac{j^2}{(j+1)!}\right) - e + \frac{3}{2} \right| + c\left|\frac{2}{M!}\right| \tag{19}$$

$$\leq 2 + c - \alpha(2e-4) + o(1). \tag{20}$$

Hence, if we set $\alpha := \frac{2+c}{2e-4}$, $|a_M - \alpha M| = |f(M)| = o(1)$.
Then, we prove that the statement holds for M.

$$\left| a_M - \frac{M}{e-2} \right| = \left| a_M - \alpha M + M(\alpha - \frac{1}{e-2}) \right| = \left| a_M - \alpha M + M\frac{c}{2e-4} \right| \tag{21}$$

$$\left| \frac{a_M}{M} - \frac{1}{e-2} \right| = \left| \frac{a_M}{M} - \alpha + \frac{c}{2e-4} \right| \leq \left| \frac{a_M}{M} - \alpha \right| + \left| \frac{c}{2e-4} \right| \leq o(\frac{1}{M}) + \frac{c}{2e-4} \leq o(1) \tag{22}$$

Therefore, for any integer $M \geq 2$, the expected number of trajectory segment pairs collected from sequential pairwise comparison a_M is approximately linear to M as $|a_M/M - 1/(e-2)| = o(1)$. $\square$

Theorem 1 denotes that $a_M/(M-\gamma)$ is approximated to $1/(e-2)$ for sufficiently large M. Substituting M for a value of greater than 100, the expected feedback efficiency of sequential pairwise comparison is calculated as $1.392(M - 1.324)/(M-1)$, which converges to 1.392 as M increases. For $M \geq 4$, the error of the linear approximation of a_M is bounded as 0.003.

B.1.2 Extension to Root Pairwise Comparison

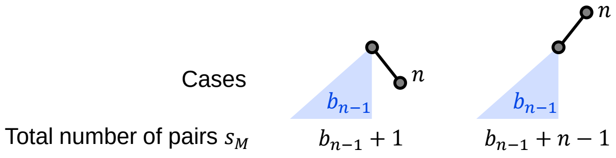


Figure 2: **Total number of pairs from two cases in root pairwise comparison.** The left figure shows the case of the n^{th} segment being preferred to the root segment until $n-1$. The right figure shows the opposite case. Numbers indicate the total number of pairs from all the cases in each scenario.

Proof of Theorem 2.

$$b_M = 2(M - \sum_{n=1}^{M} \frac{1}{n}). \tag{23}$$

(23) is satisfied for $M = 2$ as $b_2 = 1$. Intuitively, if the last pair of trajectory segments is increasing, the expected number of pairs is $b_{n-1} + n - 1$ and otherwise, the expected number of pairs is $b_{n-1} + 1$ for all $n \geq 3$. Considering that the probability of the last pair is increasing is $\frac{1}{n}$,

$$b_n = \frac{1}{n}(b_{n-1} + n - 1) + \frac{n-1}{n}(b_{n-1} + 1) = b_{n-1} + \frac{2(n-1)}{n} \tag{24}$$

is satisfied for all $n \geq 3$. Therefore, for $M \geq 2$, b_M can be written as follows:

$$b_M = b_2 + 2\sum_{n=3}^{M}(1 - \frac{1}{n}) = 2(M - \sum_{n=1}^{M}\frac{1}{n}). \tag{25}$$

□

B.2 Convergence of the Reward Model

B.2.1 High Probability Feedback Efficiency

In this section, we provide detailed proof of Lemma 2 using Lemma B.1.

***Lemma* B.1.** *Let X be an n-dimensional random variable with a dependency graph G. Let f be c-Lipschitz and c_i be a coefficient for each dimension. Assume that G consists of T-disjoint sub-graphs, i.e., $V(G) = \cup_{i=1}^{T} V(G_i)$, $E(G) = \cup_{i=1}^{T} E(G_i)$ and, for any i and j, $V(G_i) \cap V(G_j) = \emptyset$ and $E(G_i) \cap E(G_j) = \emptyset$. Then, for $\epsilon > 0$,*

$$P\left(f(X) - \mathbb{E}[f(X)] \geq \epsilon\right) \leq \exp\left(-\frac{2\epsilon^2}{\left(\max_{t\in[T]} \Delta(G_t) + 1\right)\sum_{i=1}^{n} c_i^2}\right) \tag{26}$$

***Proof of Lemma B.1**.* We simply apply the result of [1, 2] to T separated sub-graphs. First, from [1, 2], we have the following inequality,

$$P\left(f(X) - \mathbb{E}[f(X)] \geq \epsilon\right) \leq \exp\left(-\frac{2\epsilon^2}{\chi^*(G)\sum_{i=1}^{n} c_i^2}\right) \tag{27}$$

where $\chi^*(G)$ is the fractional chromatic number of G. From the fact that $\chi^*(G) \leq \Delta(G) + 1$, we have $\chi^*(G) \leq \max_{t\in[T]} \Delta(G_t) + 1$. Note that, from the assumption, G consists of the union of T separated sub-graphs. Hence, the maximum degree of G is exactly the same as the maximum of $\Delta(G_t)$ among T sub-graphs. □

***Definition* B.2.** *We formalize the definition of $S_{M,T}$ as the number of data points generated after T iterations with M comparisons per iteration.*

$$S_{M,T} := \sum_{t=1}^{T} s_{M,t} = \sum_{t=1}^{T} \sum_{1 \le i < j \le M} \mathbb{I}\left[(\sigma_{t,i}, \sigma_{t,j}) \in \mathcal{D}\right] \tag{28}$$

***Proof of Lemma 2*.** From Definition B.2, the expectation of $\mathbb{E}[S_{M,T}]$ can be computed as follows:

$$\mathbb{E}[S_{M,T}] = \sum_{t=1}^{T} \mathbb{E}[s_{M,t}] = \beta TM, \tag{29}$$

where β is the ratio of the number of train samples generated by the comparison method.

Let us define $I_{ij,t} := \mathbb{I}\left[(\sigma_{i,t}, \sigma_{j,t}) \in \mathcal{D}\right]$ and $I_{M,T}$ be a random vector that stacks $[I_{ij,t}]_{t \in [T], 1 \le i < j \le M}$. Note that $I_{M,T}$ is $TM(M-1)/2$ dimensional random variables. Then, the random variable $S_{M,T}$ can be considered as $f(I_{M,T})$ where $f(x) := \sum_i x_i$ whose Lipschitz coefficients are all 1.

From our procedure, we reset the sequence of comparisons for every iteration. Hence, data points generated in different iterations are independent while data points generated in the same iteration may be dependent on each other. In other words, our procedure generates a T-separated dependency graph. Then, from Lemma B.1,

$$P\left(S_{M,T} - \beta TM \le -\epsilon\right) \le \exp\left(-\frac{2\epsilon^2}{\left(\max_t \Delta_{M,t} + 1\right) TM(M-1)/2}\right) \ge \delta, \tag{30}$$

by setting $\epsilon = \sqrt{\left(\max_t \Delta_{M,t} + 1\right) TM(M-1)\ln(1/\delta)/4}$. If T is sufficiently large such that $T > \frac{(\max_t \Delta_{M,t}+1)(M-1)\ln(1/\delta)}{\beta^2 M}$ holds, with probability at least $1-\delta$,

$$\begin{aligned} S_{M,T} &> \beta TM - \sqrt{\frac{\left(\max_t \Delta_{M,t} + 1\right) TM(M-1)\ln(1/\delta)}{4}} \qquad (31) \\ &> \beta TM - \sqrt{\frac{TM \times \beta^2 TM}{4}} = \frac{\beta TM}{2}. \qquad (32) \end{aligned}$$

□

B.2.2 Generalization Bound with Rademacher Complexity

In this section, we provide detailed proofs of Theorem 3, Lemma 3, Lemma 4, and Corollary 3.1. We first find Rademacher bounds for a dependent dataset in Lemma B.2 and Rademacher complexity of a linear model in Lemma B.3 based on prior work [3].

***Lemma* B.2** (Rademacher bounds for dependent dataset)**.** *Given a set of samples $\mathcal{D}$ of size n with dependency graph G and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, B_\ell]$. Let $\theta^\star$ be the parameter of an oracle linear classifier. We set $\theta(x) := \theta^\intercal x$ for simplicity. Then, for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\forall \theta \in \mathcal{H}, \;\; R(\theta) \le \hat{R}(\theta) + 2\hat{\mathfrak{R}}_{\mathcal{D}}(\ell \circ \theta) + 3B_\ell \sqrt{\frac{\chi_f(G)}{2n} \ln\left(\frac{4}{\delta}\right)}, \tag{33}$$

where $\mathcal{H}$ is the parameter space of θ, B_ℓ is a bound of loss function ℓ, and $\hat{\mathfrak{R}}_{\mathcal{D}}(\ell \circ \theta)$ is the empirical Rademacher complexity of $\ell \circ \theta$. If G consists of T-separated sub-graphs, then,

$$\forall \theta \in \mathcal{H}, \;\; R(\theta) \le \hat{R}(\theta) + 2\hat{\mathfrak{R}}(\ell \circ \theta) + 3B_\ell \sqrt{\frac{\max_t \Delta(G_t) + 1}{2n} \ln\left(\frac{4}{\delta}\right)} \tag{34}$$

***Lemma* B.3** (Rademacher complexity of linear model on dependency graph G)**.** *Let $\|\theta\|_2 \le B_\theta$ and $\|x\|_2 \le B_x$. Let $\ell(x)$ be a 1-Lipschitz function.*

$$\forall \theta \in \mathcal{H}, \;\; \hat{\mathfrak{R}}_S(\ell \circ \theta) \le B_\theta B_x \sqrt{\frac{\chi_f(G)}{n}}, \tag{35}$$

where B_θ is the upper bound of the norm of θ and B_x is the upper bound of the norm of x. If G consists of T-separated sub-graphs, then,

$$\forall \theta \in \mathcal{H}, \quad \hat{\mathfrak{R}}_S(\ell \circ \theta) \leq B_\theta B_x \sqrt{\frac{\max_t \Delta(G_t) + 1}{n}} \tag{36}$$

Proof of Lemma B.2 and B.3. (33) and (35) hold from the results of [3]. Then, for T-separated sub-graphs, we can apply $\chi_f(G) \leq \Delta(G) + 1 \leq \max_t \Delta(G_t) + 1$. ☐

We find the bound of reward model parameter θ as the following lemma.

***Lemma* B.4** (Boundedness of the parameter in logistic regression)**.** *Suppose that M segments are compared per iteration, and T iterations are executed. Let $\hat{\theta}$ be the minimizer of the logistic regression with a regularization coefficient $\lambda > 0$ as follows,*

$$\min_\theta \mathcal{L}(\theta) = \sum_{(x_i, y_i) \in \mathcal{D}} \ln\left(1 + e^{-y_i x_i^\intercal \theta}\right) + \frac{\lambda}{2}\|\theta\|_2^2 \tag{37}$$

Then, $\|\hat{\theta}\|_2 \leq \frac{2SM(M-1)}{\kappa \lambda_{\min}(\Sigma_X)} + 2D$ holds with probability at least $1-\delta$ for $T \geq 2\left(\frac{C_1\sqrt{d} + C_2\sqrt{\ln(1/\delta)}}{\lambda_{\min}(\Sigma_X)}\right)^2$, where $C_1 =$, $C_2 =$, $\lambda_{min} =$, $\Sigma_X =$, and $\kappa = \inf_{\|\theta-\theta^\star\|_2 \leq D, \|x\|_2 \leq 2S} \frac{e^{-x^\intercal\theta}}{\left(1+e^{-x^\intercal\theta}\right)^2}$.

Proof of Lemma B.4. From the fact that $\hat{\theta}$ is the minimizer of $\mathcal{L}$, we have,

$$\nabla_\theta \mathcal{L}(\hat{\theta}) = \sum_{(x_i, y_i) \in \mathcal{D}} (p_i(\hat{\theta}) - 1) y_i x_i + \lambda \hat{\theta} = 0, \tag{38}$$

where $p_i(\hat{\theta}) := (1 + e^{-y_i x_i^\intercal \hat{\theta}})^{-1}$. The Hessian of $\mathcal{L}$ is computed as

$$\nabla_\theta^2 \mathcal{L}(\theta) = \sum_{(x_i, y_i) \in \mathcal{D}} p_i(\theta)[1 - p_i(\theta)] y_i^2 x_i x_i^\intercal + \lambda \mathbb{I}_d. \tag{39}$$

If $\|\theta - \theta^\star\|_2 \leq r$, let us define κ_r as $\inf_{\|\theta-\theta^\star\|_2 \leq r, \|x_i\|_2 \leq 2S} p_i(\theta)[1 - p_i(\theta)]$. Note that data samples generated from different iterations are independent. Hence, we can select at least one sample per iteration and have T i.i.d. samples in total.

$$\begin{aligned}
\nabla_\theta^2 \mathcal{L}(\theta) &\succ \kappa_r \sum_{(x_i, y_i) \in \mathcal{D}} y_i^2 x_i x_i^\intercal + \lambda \mathbb{I}_d \qquad (40) \\
&\succ \kappa_r \sum_{t=1}^{T} x_t x_t^\intercal + \lambda \mathbb{I}_d \succ \left(\kappa_r \lambda_{\min}(T) + \lambda\right) \mathbb{I}_d, \qquad (41)
\end{aligned}$$

where $\lambda_{\min}(T)$ is the smallest eigen value of $\sum_{t=1}^{T} x_t x_t^\intercal$. From the mean value theorem, the following equality holds for $\bar{\theta}$ such that $\bar{\theta} = c\hat{\theta} + (1-c)\theta^\star$ for $c \in (0, 1)$.

$$\begin{aligned}
\|\nabla_\theta \mathcal{L}(\theta^\star)\|_2 &= \left\|\nabla_\theta \mathcal{L}(\theta^\star) - \nabla_\theta \mathcal{L}(\hat{\theta})\right\|_2 = \left\|\nabla_\theta^2 \mathcal{L}(\bar{\theta})[\theta^\star - \hat{\theta}]\right\|_2 \qquad (42) \\
&\geq \left(\kappa_r \lambda_{\min}(T) + \lambda\right) \|\theta^\star - \hat{\theta}\|_2 \qquad (43) \\
\|\nabla_\theta \mathcal{L}(\theta^\star)\|_2 &\leq \sum_{(x_i, y_i) \in \mathcal{D}} |y_i(p_i(\theta^\star) - 1)| \|x_i\|_2 + \lambda \|\theta^\star\|_2 \qquad (44) \\
&\leq 2S \cdot S_{M,T} + \lambda D \leq 2S \cdot TM(M-1)/2 + \lambda D \qquad (45) \\
\|\theta^\star - \hat{\theta}\|_2 &\leq \frac{STM(M-1) + \lambda D}{(\kappa_r \lambda_{\min}(T) + \lambda)} \leq \frac{STM(M-1) + \lambda D}{\kappa_r \lambda_{\min}(\Sigma_X) T/2 + \lambda} \leq \frac{2SM(M-1)}{\kappa_r \lambda_{\min}(\Sigma_X)} + D \qquad (46)
\end{aligned}$$

Hence, we have $\|\theta^\star - \hat{\theta}\|_2 \leq \frac{2SM(M-1)}{\kappa_r \lambda_{\min}(\Sigma_X)} + D$. Then, for any dataset, the norm of $\hat{\theta}$ can be bounded as $\|\hat{\theta}\|_2 \leq \|\hat{\theta} - \theta^\star\|_2 + \|\theta^\star\|_2 \leq \frac{2SM(M-1)}{\kappa_r \lambda_{\min}(\Sigma_X)} + 2D$. Then, if we set r to be greater than this bound, then, we can define κ such that $\frac{2SM(M-1)}{\kappa \lambda_{\min}(\Sigma_X)} + D \leq r$ and $\kappa_r \leq \kappa$ hold. ☐

By applying Lemma B.4 and Lemma B.3 to Lemma B.2, we can prove Theorem 3 as follows.

Proof of Theorem 3. Let θ be the minimizer of the logistic regression with a regularization coefficient $\lambda > 0$. Then, for $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following bound holds,

$$\forall \theta \in \mathcal{H}, \;\; R(\theta) \le \hat{R}(\theta) + \left(\frac{2SM(M-1)}{\kappa \lambda_{\min}(\Sigma_X)} + 2D \right) 2S \sqrt{\frac{\max_t \Delta_{M,t} + 1}{S_{M,T}}} \tag{47}$$

$$+ 3 \ln \left(1 + \exp \left(\frac{2S^2 M(M-1)}{\kappa \lambda_{\min}(\Sigma_X)} + 2SD \right) \right) \sqrt{\frac{\max_t \Delta_{M,t} + 1}{2S_{M,T}} \ln \left(\frac{4}{\delta} \right)} \tag{48}$$

Therefore, we can derive the statement as follows:

$$\forall \theta \in \mathcal{H}, \;\; R(\theta) \le \hat{R}(\theta) + \sqrt{\frac{\max_t \Delta_{M,t} + 1}{S_{M,T}}} \cdot \left(2FD + 3B \sqrt{\frac{1}{2} \ln \left(\frac{4}{\delta} \right)} \right), \tag{49}$$

where $F = \left(\frac{2SM(M-1)}{\kappa D \lambda_{\min}(\Sigma_X)} + 2 \right) S = \mathcal{O}(M^2)$ and $B = \ln \left(1 + \exp \left(\frac{2S^2 M(M-1)}{\kappa \lambda_{\min}(\Sigma_X)} + 2SD \right) \right) = \mathcal{O}(M^2)$. □

As we address in the paper, a trade-off exists between the feedback efficiency and data dependency in (49). To demonstrate $\Delta_{M,t}$, we prove Lemma 3 and 4 for sequential and root pairwise comparison, respectively.

Proof of Lemma 3. Let us first consider the degree Δ_M of the dependency graph G at a single iteration. Let K be the length of the longest fully increasing or decreasing subsequence $(\sigma_i, ..., \sigma_{i+K-1})$, where i is the index of the first node in this sequence. Let I_1 and I_2 be the maximum length of fully increasing or decreasing subsequence that ends with σ_i and that starts from σ_{i+K-1}, respectively. Hence, the maximum degree of the entire graph G has the following relationship.

$$\Delta_M = \max(K + I_1 - 3, K + I_2 - 3) = J + \max(I_1, I_2) - 3 \tag{50}$$

Then for a constant c and a sufficiently large M,

$$P(\Delta_M \le c) \sum_{k=2}^{M-1} P(J + \max(I_1, I_2) - 3 \le c | K = k) P(K = k) \tag{51}$$

$$\ge \sum_{k=2}^{M-1} P(2K - 3 \le c | K = k) P(K = k) \tag{52}$$

$$\ge \sum_{k=2}^{M-1} P(2k - 3 \le c) P(K = k) \tag{53}$$

$$\ge \sum_{k=2}^{\lfloor \frac{c+3}{2} \rfloor} P(K = k) \tag{54}$$

$$= P(K \le \frac{c+3}{2}) \tag{55}$$

$$= 1 - P(K > \frac{c+3}{2}) \tag{56}$$

$$= 1 - \frac{\binom{M}{\frac{c+3}{2}} \cdot 2 \cdot (M - \frac{c+3}{2} + 1) \cdot (M - \frac{c+3}{2})!}{M!} \tag{57}$$

$$= 1 - \frac{2(M - \frac{c+3}{2} + 1)}{\frac{c+3}{2}!} \ge \exp \left(-\frac{M}{e^{c-2} - 1} \right) \ge 1 - \delta. \tag{58}$$

Hence, if $c := 2 + \ln(1 + M/(\ln(1/(1-\delta))))$, $\Delta_M \leq 2 + \ln(1 + M/(\ln(1/(1-\delta))))$ holds with probability at least $1 - \delta$.

For all $t \in [T]$, with probability at least $1 - \delta$, the maximum degree of dependency graph has the following upper bound.

$$\max_{t\in[T]} \Delta_{M,t} \leq 2 + \ln(1 + M/(\ln(1/(1-\delta)))) \tag{59}$$

□

Proof of Lemma 4. Let us first consider the degree Δ_M of the dependency graph G at a single iteration. In root pairwise comparison, the agent constructs a tree structure of segments $\mathcal{T}$. Suppose $\mathcal{D}_{nl}(\mathcal{T}) = \{\sigma_0^*, ..., \sigma_L^*\}$ is the set of non-leaf nodes in $\mathcal{T}$, where L and σ_i^* denote the depth of $\mathcal{T}$ and the i^{th} non-leaf node inserted to $\mathcal{T}$, respectively. In other words, σ_L^* denotes the root node of $\mathcal{T}$ after the tree is fully constructed. After augmenting queries, edges are added for every pair of parent-child nodes. Let $\mathcal{T}'$ be the augmented tree. The maximum degree of the dependency graph is achieved as the degree of $(\sigma_{L-1}^*, \sigma_L^*)$ in the dependency graph G, as both nodes in this query have the most children among nodes in $\mathcal{T}'$. Therefore, we obtain Δ_M as follows:

$$\begin{aligned}\Delta_M &= d_{\mathcal{T}'}(\sigma_L^*) + d_{\mathcal{T}'}(\sigma_{L-1}^*) - 2 &(60)\\ &= (M-1) + (M - d_{\mathcal{T}}(\sigma_L^*)) - 2 &(61)\\ &= 2M - d_{\mathcal{T}}(\sigma_L^*)) - 3, &(62)\end{aligned}$$

where $d_{\mathcal{T}'}(\sigma)$ denotes the degree of a segment σ in the augmented tree $\mathcal{T}'$. Then for a constant c,

$$\begin{aligned}P(\Delta_M \leq c) &= P(2M - d_{\mathcal{T}}(\sigma_L^*)) - 3 \leq c) &(63)\\ &= P(d_{\mathcal{T}} \geq 2M - 3 - c) &(64)\\ &= P(i \leq M - (2M - 3 - c) \; s.t. \; \sigma_i = \sigma_L^*) &(65)\\ &= \frac{-M + 3 + c}{M} \geq 1 - \delta, &(66)\end{aligned}$$

where σ_i denotes the i^{th} sampled trajectory segment.

Hence, if $c := M(2-\delta) - 3$, $\Delta_M \leq M(2-\delta) - 3$ holds with probability at least $1 - \delta$.

For all $t \in [T]$, with probability at least $1 - \delta$, the maximum degree of dependency graph has the following upper bound.

$$\max_{t\in[T]} \Delta_{M,t} \leq M(2-\delta) - 3 \tag{67}$$

□

Proof of Corollary 3.1. By applying Lemma 3 and 4 to Theorem 3, we derive the generalization bounds of the reward model using sequential and root pairwise comparison, respectively. For pairwise comparison, we substitute $\max_t \Delta_{M,t} = 0$ and $S_{M,T} = TM$ in Theorem 3. Note that F and B are constants as we fix M from the assumption. By substituting $\delta = 1/T$, we prove the generalization bound for each trajectory comparison method as follows.

Pairwise comparison:

$$\begin{aligned}\forall\theta \in \mathcal{H}, \;\; R(\theta) &\leq \hat{R}(\theta) + \sqrt{\frac{1}{TM}} \cdot \left(2FD + 3B\sqrt{\frac{1}{2}(\ln(T) + \ln(4))}\right) &(68)\\ &= \hat{R}(\theta) + \mathcal{O}\left(\sqrt{\frac{\ln(T)}{T}}\right) &(69)\end{aligned}$$

Sequential pairwise comparison:

$$\forall\theta \in \mathcal{H}, \;\; R(\theta) \leq \hat{R}(\theta) + \sqrt{\frac{3 + \ln\left(1 + \frac{M}{\ln(\frac{1}{1-\frac{1}{T}})}\right)}{1.392TM}} \cdot \left(2FD + 3B\sqrt{\frac{1}{2}(\ln(T) + \ln(4))}\right) \tag{70}$$

$$= \hat{R}(\theta) + \mathcal{O}\left(\sqrt{\ln\left(\frac{1}{\ln(\frac{1}{1-\frac{1}{T}})}\right)\frac{\ln(T)}{T}}\right) \tag{71}$$

$$= \hat{R}(\theta) + \mathcal{O}\left(\sqrt{\frac{(\ln(T))^2}{T}}\right) \quad \because \text{linear approximation for } T \gg 1 \tag{72}$$

Root pairwise comparison:

$$\forall\theta \in \mathcal{H}, \quad R(\theta) \leq \hat{R}(\theta) + \sqrt{\frac{M(2-1/T)-3+1}{2TM}} \cdot \left(2FD + 3B\sqrt{\frac{1}{2}(\ln(T)+\ln(4))}\right) \tag{73}$$

$$= \hat{R}(\theta) + \mathcal{O}\left(\sqrt{\frac{\ln(T)}{T}}\right) \quad \because \text{for sufficiently large } T,\ 1/T \to 0 \tag{74}$$

Therefore, we can find the convergence rates of generalization bounds of the reward model as $\mathcal{O}(\sqrt{\ln(T)/T})$, $\mathcal{O}(\sqrt{(\ln(T))^2/T})$, and $\mathcal{O}(\sqrt{\ln(T)/T})$ with probability at least $1-1/T$ for pairwise, sequential pairwise, and root pairwise comparison, respectively. □

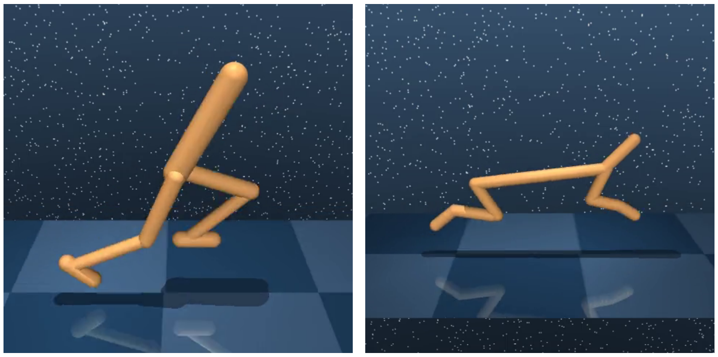
C Algorithm

Algorithm 1 summarizes the process of RLHF using sequential pairwise comparison. Algorithm 2 summarizes the process of RLHF using root pairwise comparison.

```
Algorithm 1: Sequential Pairwise Comparison RLHF
Require: Teacher frequency $K$, queries per session $N$, time horizon of a trajectory: $T$
time horizon of a segment: $H$
Learning rate of policy update: $\nu_\phi$, learning rate of reward update: $\nu_\theta$
Initialize parameters of $\pi_\phi$ and $\hat{r}_\theta$
Initialize buffers $\mathcal{U} \leftarrow \emptyset$
for each iteration do
    // SIMULATION
    for $n = 1, ..., N_s$ do
        for each timestep $t = 1, ..., T$ do
            Collect $s_{t+1}$ by taking $a_t \sim \pi_\phi(a_t|s_t)$
        end
        $\mathcal{U} \leftarrow \mathcal{U} \cup \{\tau_n|\tau_n = (s_1, a_1, ..., s_T, a_T)\}$
    end
    // POLICY LEARNING
    for each gradient step do
        Sample random minibatch $\mathcal{B} = \{\tau_j\}_{j=1}^{B} \sim \mathcal{U}$
        $\phi \leftarrow \phi - \nu_\phi \nabla_\phi L_{policy}(\phi, \mathcal{B})$
    end
    // REWARD LEARNING
    Initialize preference buffer $\mathcal{D} \leftarrow \emptyset$
    Randomly sample $\sigma_1 \in \mathcal{U}$
    for $m = 1, 2, .., N$ do
        Randomly sample the $m^{th}$ challenger $\sigma_{m+1} \in \mathcal{U} \setminus \{\sigma_1, ..., \sigma_m\}$
        $y_m \leftarrow$ true preference label for pair $(\sigma_m, \sigma_{m+1})$
        $\mathcal{D} \leftarrow \mathcal{D} \cup ((\sigma_m, \sigma_{m+1}), y_m)$
    end
    // Augment pairs by sequential preference ranking
    $inc \leftarrow (\sigma_1 \prec \sigma_2)$
    Initialize temporal buffer $\mathcal{K} \leftarrow \{\sigma_1\}$
    for $m = 2, .., N$ do
        if $inc$ is not $(\sigma_m \prec \sigma_{m+1})$ then
            for $i < j$ s.t. $\sigma_i, \sigma_j \in \mathcal{K}$ do
                $\mathcal{D} \leftarrow \mathcal{D} \cup ((\sigma_i, \sigma_j), \neg inc)$
            end
            Reinitialize $\mathcal{K} \leftarrow \{\sigma_{m+1}\}$
            Reinitialize $inc \leftarrow (\sigma_m \prec \sigma_{m+1})$
        else
            $\mathcal{K} \leftarrow \mathcal{K} \cup \{\sigma_{m+1}\}$
        end
    end
    $\theta \leftarrow \theta - \nu_\theta \nabla_\theta L_{pref}(\theta, \mathcal{D})$
end
```

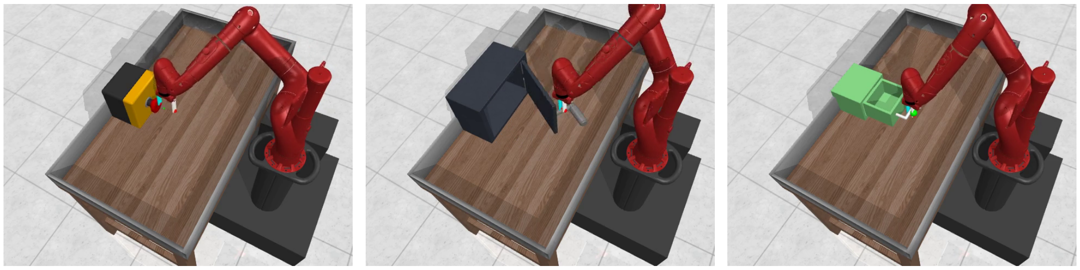
Algorithm 2: Root Pairwise Comparison RLHF

```
Require: Teacher frequency $K$, queries per session $N$, time horizon of a trajectory: $T$
time horizon of a segment: $H$
Learning rate of policy update: $\nu_\phi$, learning rate of reward update: $\nu_\theta$
Initialize parameters of $\pi_\phi$ and $\hat{r}_\theta$
Initialize buffers $\mathcal{U} \leftarrow \emptyset$
for each iteration do
    // SIMULATION
    for $n = 1, ..., N_s$ do
        for each timestep $t = 1, ..., T$ do
            Collect $s_{t+1}$ by taking $a_t \sim \pi_\phi(a_t|s_t)$
        end
        $\mathcal{U} \leftarrow \mathcal{U} \cup \{\tau_n | \tau_n = (s_1, a_1, ..., s_T, a_T)\}$
    end
    // POLICY LEARNING
    for each gradient step do
        Sample random minibatch $\mathcal{B} = \{\tau_j\}_{j=1}^{B} \sim \mathcal{U}$
        $\phi \leftarrow \phi - \nu_\phi \nabla_\phi L_{policy}(\phi, \mathcal{B})$
    end
    // REWARD LEARNING
    Initialize preference buffer as a tree $\mathcal{T} \leftarrow \emptyset$
    Randomly sample $\sigma_1 \in \mathcal{U}$
    $Root(\mathcal{T}) \leftarrow \sigma_1$
    for $m = 1, 2, .., N$ do
        Randomly sample $\sigma_{m+1} \in \mathcal{U} \setminus \{\sigma_1, ..., \sigma_m\}$
        $y_m \leftarrow$ true preference label for pair $(\sigma_m, \sigma_{m+1})$
        if $y_m$ is $0$ then
            // Insert $\sigma_{m+1}$ as root
            $Root(\mathcal{T}) \leftarrow \sigma_{m+1}$
        else
            // Insert $\sigma_{m+1}$ as a new child of current root
            $Root(\mathcal{T}).newChild \leftarrow \sigma_{m+1}$
        end
    end
    // Augment pairs by sequential preference ranking
    for $\sigma_i \in \mathcal{T}$ do
        for $\sigma_j \in \mathcal{T}$ s.t. $\sigma_j$ is a child of $\sigma_i$ do
            $\mathcal{D} \leftarrow \mathcal{D} \cup ((\sigma_i, \sigma_j), +1)$
        end
    end
    $\theta \leftarrow \theta - \nu_\theta \nabla_\theta L_{pref}(\theta, \mathcal{D})$
end
```

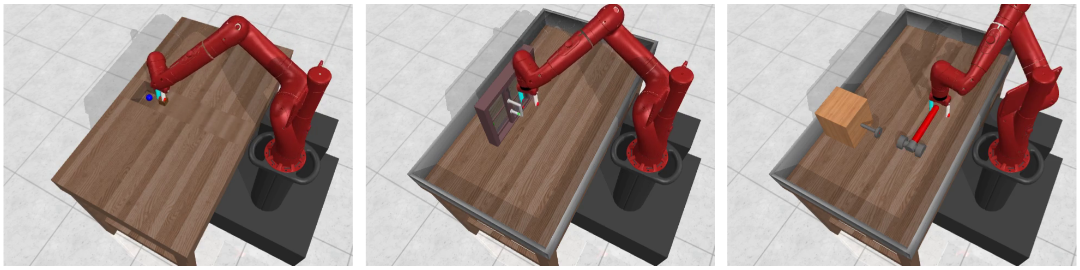


(a) Walker Walk (b) Cheetah Run

(c) Quadruped Walk (d) Humanoid Walk (e) Hopper Hop



(f) Button Press (g) Door Open (h) Drawer Open



(i) Sweep Into (j) Window Open (k) Hammer

Figure 3: **Visualization of Tasks.**

D Experiments

D.1 Experiment Settings

Tasks. We evaluate our algorithm in locomotion tasks from DMControl [4, 5] and manipulation tasks from Meta-World [6] as shown in Figure 3.

Baselines. We use both MRN[7] and PEBBLE[8] as baselines of the proposed methods. Experiment results with PEBBLE baseline are provided in Section D.5.

Computation. We train and evaluate each task in DMControl and Meta-World on a single machine with one NVIDIA GeForce RTX 3090 GPU for 160 hours. For the block placing task, we pre-train the agent in the mujoco simulator on a single machine with one NVIDIA GeForce RTX 3090 GPU for 16 hours and fine-tune the agent in the real-world on a single machine with one NVIDIA GeForce RTX 2060 GPU for 1.5 hours.

Experiments with real human feedback. A total of 35 individuals (8 female, 27 male) between the ages of 21 and 61 were selected for our study. All participants had prior knowledge of the concept of reinforcement learning and the importance of appropriate rewards in agent training. Each participant provided feedback based on their preferences. Then, the policy and reward models were fine-tuned using the feedback data. The random seed was fixed as 12345.

Real robot experiments. The tabletop environment has a size of 60 cm x 70 cm. The initial position of the block is randomly sampled from the 20 cm x 20 cm space on the top left corner of the tabletop plane. Then, the agent has to pick and place the block at the fixed goal position, located in the bottom right corner of the tabletop plane. At each action, the agent can move the block for $5cm$ in a direction chosen from $\{0, \frac{\pi}{4}, \frac{2\pi}{4}, .., \frac{7\pi}{4}\}$. The episode is considered a success if the agent moves the block's center of mass near the goal at a distance less than $3cm$. During the pre-training phase, we start with $\epsilon = 1.0$ to collect diverse trajectories for 1000 steps. Then, we gradually reduce ϵ from 0.5 to 0.1 with a rate of 0.98 per 100 steps during the learning phase. For fine-tuning in the real world, we collect diverse trajectories with $\epsilon = 0.3$ for 100 steps. Then, we gradually reduce ϵ with a rate of 0.98. Considering the training time, we fix the random seed as 12345.

D.2 Implementation Details

The length of a segment is fixed as $H = 50$, and the learning rate of the reward model is fixed as $3e-4$. For each task, we perform unsupervised pre-training [8] for $9,000$ steps on each model. Table 2 and Table 3 show training hyperparameters, including the learning rate, frequency of feedback, the maximum number of feedback, the batch size for reward updates, the meta update frequency [7], and the number of training steps for simulation experiments.

Tasks	learning rate	feedback frequency	max feedback	reward batch size	meta update frequency	training steps
Walker Walk	5e-4	20,000	400	40	1,000	5e5
Cheetah Run	5e-4	20,000	200	20	1,000	1e6
Quadruped Walk	1e-4	30,000	1,000	100	1,000	1e6
Humanoid Walk	1e-4	30,000	40,000	500	10,000	1e6
Hopper Hop	1e-4	20,000	4,000	100	1,000	1e6
Button Press	3e-4	5,000	10,000	50	5,000	1e6
Door Open	3e-4	5,000	10,000	200	10,000	1e6
Drawer Open	3e-4	5,000	20,000	200	5,000	1e6
Sweep Into	3e-4	5,000	10,000	100	5,000	1e6
Window Open	3e-4	5,000	1,000	100	5,000	1e6
Hammer	3e-4	5,000	10,000	30	10,000	1e6

Table 2: **Hyperparameters for Training in DMControl and Meta-World Tasks.**

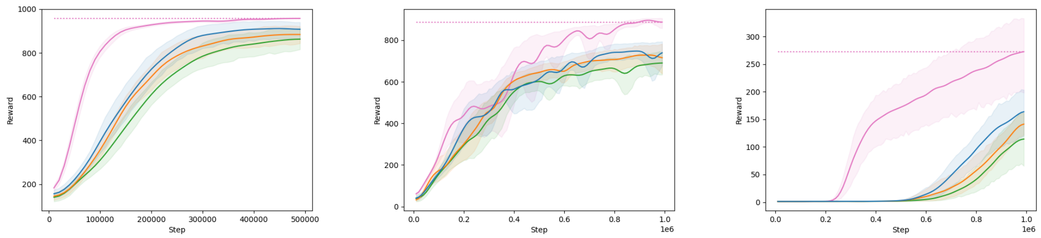
Tasks	learning rate	feedback frequency	max feedback	reward batch size	meta update frequency	training steps
Cheetah Run (Real human feedback)	1e-8	200	20	10	-	400

Table 3: **Hyperparameters for Training the Cheetah Run Task from Real Human Feedback.**

We adopt DQN from prior work [9] for real robot experiments and use hyparameters described in Table 4. The length of a segment is fixed as 30.

Tasks	learning rate of policy update	learning rate of reward update	feedback frequency	max feedback	reward batch size	training steps
Block Placing	1e-6	1e-6	200	800	50	3,000

Table 4: **Hyperparameters for Training the Block Placing Task Using the Real UR-5 Robot.**



(a) Walker Walk
(feedback=0.4K)

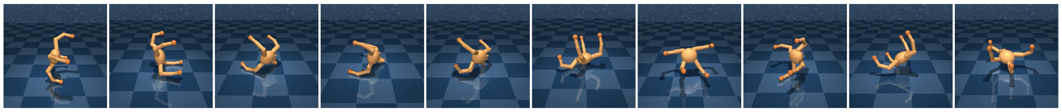
(b) Cheetah Run
(feedback=0.2K)

(c) Humanoid Walk
(feedback=40K)

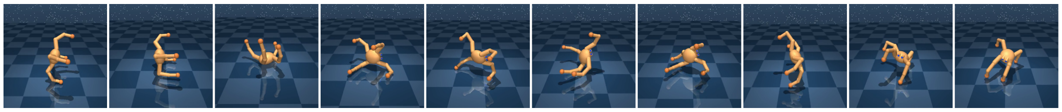
Figure 4: **Reward graphs in three locomotion tasks from DMControl.** All methods are implemented based on MRN. Each subfigure describes the comparison results among three trajectory comparison methods: pairwise (green), sequential pairwise (orange), and root pairwise (blue). Pink lines describe the oracle performance using SAC with the true reward.

D.3 Experiments in Locomotion Tasks (DMControl)

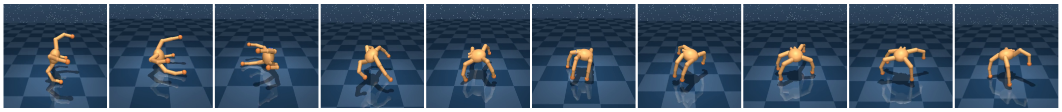
Reward graphs in Figure 4 demonstrate that the proposed sequential and root pairwise comparison outperforms the pairwise comparison baseline for walker walk, cheetah run, and humanoid walk tasks in DMControl. As we discuss in the paper, while both proposed methods are successful, root pairwise comparison shows faster convergence and higher reward after convergence compared to sequential pairwise comparison. Figure 5 and 6 illustrate the examples of agent trajectories in quadruped walk and walker walk tasks after training, respectively. In Figure 5, the agent trained using pairwise comparison fails to turn its body upside down to a normal state that the agent stands upright. On the other hand, both sequential and root pairwise comparison trains the agent to flip its body, but sequential pairwise comparison requires twice the time as root pairwise comparison. In Figure 6, while the agents trained using pairwise and sequential pairwise comparison fall and put their knees to the floor at initial timesteps, the agent trained using root pairwise comparison learns to walk by repeating bending and releasing its legs appropriately.



(a) Pairwise (Reward=112.2)

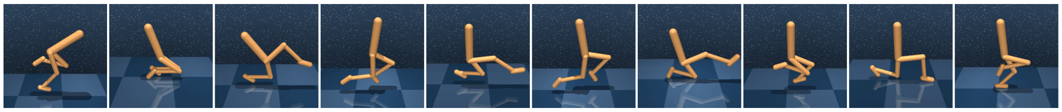


(b) Sequential Pairwise (Reward=457.7)

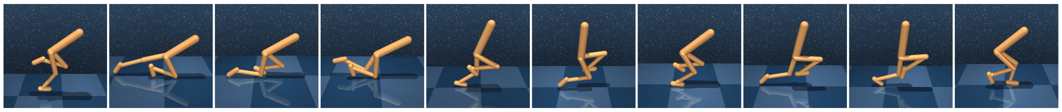


(c) Root Pairwise (Reward=934.0)

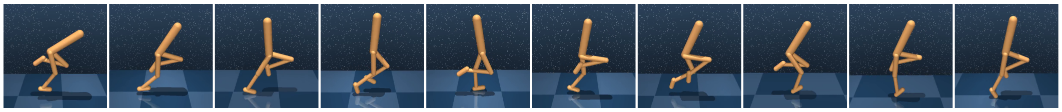
Figure 5: **Example trajectories of the agent in the quadruped walk task.** (a), (b), and (c) describe a trajectory of an agent trained using pairwise, sequential pairwise, and root pairwise comparison, respectively. The images are shown every 20 frames from the initial frame.



(a) Pairwise (Reward=730.2)

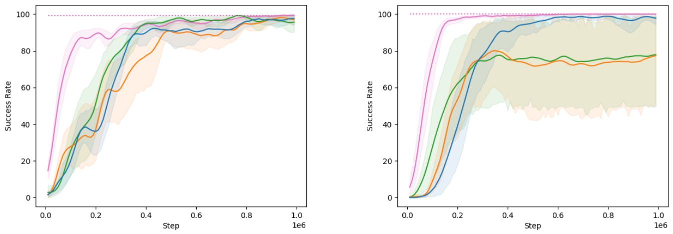


(b) Sequential Pairwise (Reward=820.9)

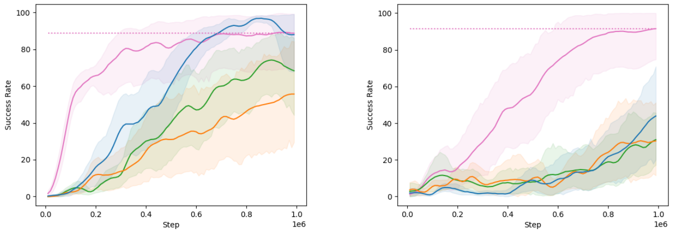


(c) Root Pairwise (Reward=985.1)

Figure 6: **Example trajectories of the agent in the walker walk task.** (a), (b), and (c) describe a trajectory of an agent trained using pairwise, sequential pairwise, and root pairwise comparison, respectively. The images are shown every 20 frames from the initial frame.



(a) Button Press (feedback=10K) (b) Door Open (feedback=10K)

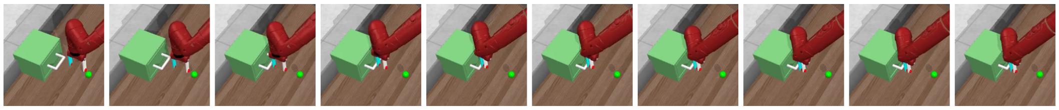


(c) Sweep Into (feedback=10K) (d) Hammer (feedback=10K)

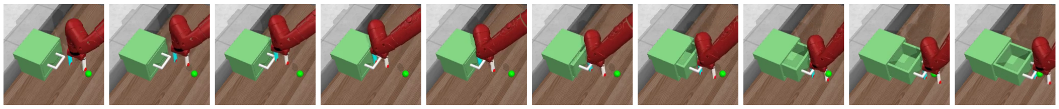
Figure 7: **Success rate graphs in four manipulation tasks from Meta-World.** All methods are implemented based on MRN. Each subfigure describes the comparison results among three trajectory comparison methods: pairwise (green), sequential pairwise (orange), and root pairwise (blue). Pink lines describe the oracle performance using SAC with the true reward.

D.4 Experiments in Manipulation Tasks (Meta-world)

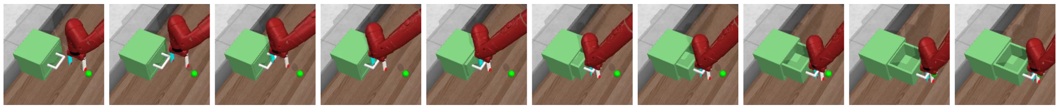
Success rate graphs in Figure 7 demonstrate that the proposed sequential and root pairwise comparison outperforms the pairwise comparison baseline for button press, door open, sweep into, and hammer tasks in Meta-World. Figure 8, 9, and 10 illustrate examples of agent trajectories in drawer open and window open tasks. Especially for the drawer open task, we analyze two different scenarios in Figure 8 and 9. Figure 8 demonstrates a scenario that the agent trained using pairwise comparison fails to open the drawer, while agents trained using sequential and root pairwise comparison succeed. Additionally, the agent trained using root pairwise comparison opens the drawer faster than the agent trained with sequential pairwise comparison. These results imply that three trajectory comparison methods are successful in the order of root pairwise, sequential pairwise, and pairwise comparison. We also analyze another scenario described in Figure 9. In this scenario, the agent trained using



(a) Pairwise (Reward=2578.1)

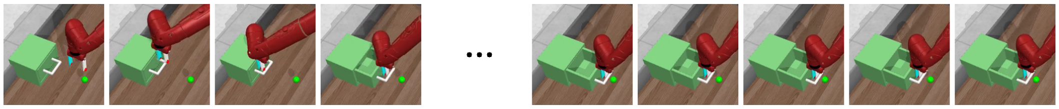


(b) Sequential Pairwise (Reward=4192.8)

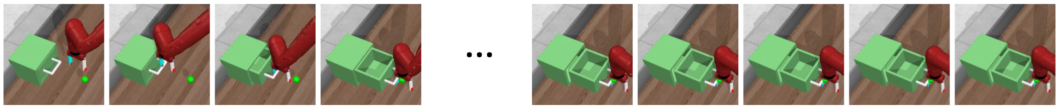


(c) Root Pairwise (Reward=4718.0)

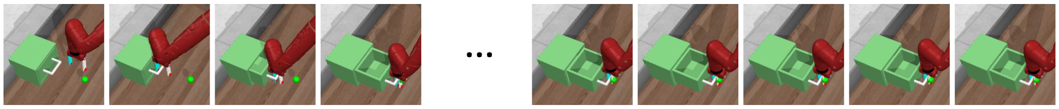
Figure 8: **Example trajectories of the agent in the drawer open task (Scenario 1).** (a), (b), and (c) describe a trajectory of an agent trained using pairwise, sequential pairwise, and root pairwise comparison, respectively. The images are shown every 5 frames from the initial frame. In this scenario, the agent trained using pairwise comparison fails to open the drawer, while agents trained using sequential and root pairwise comparison succeed.



(a) Pairwise (Reward=3880.0)



(b) Sequential Pairwise (Reward=4196.6)

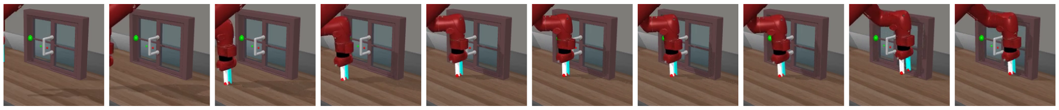


(c) Root Pairwise (Reward=4766.9)

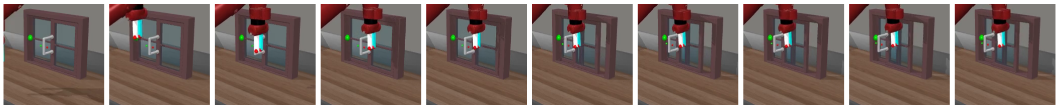
Figure 9: **Example trajectories of the agent in the drawer open task (Scenario 2).** (a), (b), and (c) describe a trajectory of an agent trained using pairwise, sequential pairwise, and root pairwise comparison, respectively. Four images on the left are shown every 5 frames from the initial frame and five images on the right are shown every 70 frames from frame 140. In this scenario, all agents trained using three different trajectory comparison methods succeed. However, the agent trained using pairwise comparison shows an unstable performance due to the oscillation of the position of the end effector after opening the drawer.

pairwise comparison succeeds in the episode but achieves a lower reward of 3880.0 compared to the reward values of 4196.6 and 4766.9 obtained from agents trained using sequential and root pairwise comparison during the episode, respectively. Comparing the rendered trajectories, the agent trained using pairwise comparison repeats to open and close the drawer slightly after pulling the drawer to the goal position. Figure 10 illustrates a scenario that the agent trained using pairwise comparison fails to open the window, while the agents trained using sequential and root pairwise comparison open the door partially and fully, respectively.

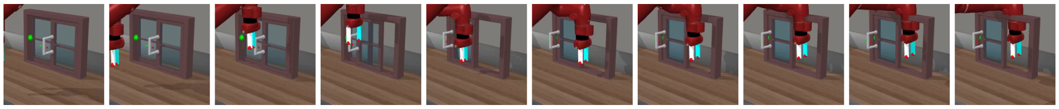
Results in Table 5 show the comparison results with baseline in terms of rewards after convergence. Root pairwise comparison shows the highest rewards on average among three trajectory comparison methods. Additionally, while sequential pairwise comparison outperforms pairwise comparison by only 1.2% in success rates, the rewards after convergence are improved by 5.2% compared to pairwise comparison. This result implies that even if the success rates are similar between pairwise comparison and sequential pairwise comparison, agents trained using sequential pairwise comparison generate more ideal actions. Reward graphs in Figure 11 also show that sequential pairwise comparison converges to higher rewards than pairwise comparison for all tasks except the sweep into task.



(a) Pairwise (Reward=288.3)



(b) Sequential Pairwise (Reward=762.3)

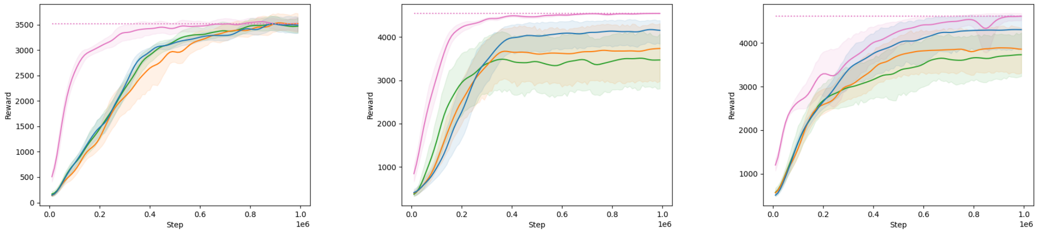


(c) Root Pairwise (Reward=853.6)

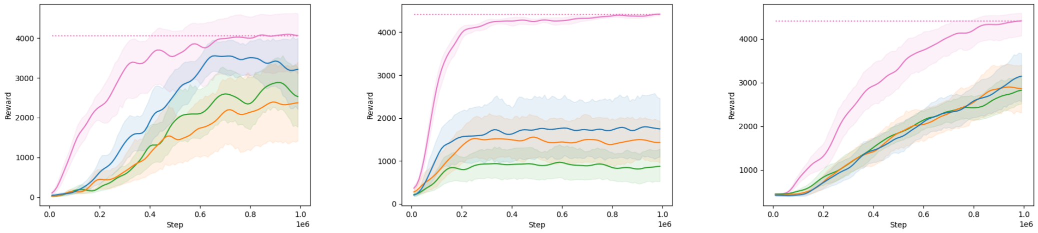
Figure 10: **Example trajectories of the agent in the window open task.** (a), (b), and (c) describe a trajectory of an agent trained using pairwise, sequential pairwise, and root pairwise comparison, respectively. The images are shown every 5 frames from the initial frame.

Task	# feedback	Oracle	Pairwise	Sequential Pairwise	Root Pairwise
Button Press	10K	3521.2 ± 229.9	3467.8 ± 254.4	**3519.1** ± 324.0	3490.0 ± 281.8
Door Open	10K	4541.1 ± 11.1	3468.0 ± 1069.4	3733.3 ± 1112.3	**4151.8** ± 460.2
Drawer Open	20K	4617.1 ± 144.7	3734.4 ± 860.2	3859.7 ± 896.0	**4309.9** ± 680.5
Sweep Into	10K	4062.9 ± 1388.1	2528.9 ± 1239.9	2371.5 ± 1667.2	**3212.7** ± 1501.7
Window Open	1K	4418.4 ± 51.8	876.8 ± 633.4	1429.5 ± 832.8	**1748.2** ± 1214.7
Hammer	10K	4415.9 ± 532.5	2824.9 ± 529.7	2862.4 ± 904.1	**3146.7** ± 943.2

Table 5: **Rewards after convergence in Meta-World manipulation tasks.** Two elements in each cell denote the average value and standard deviation of rewards after convergence across runs with 10 random seeds.



(a) Button Press (feedback=10K) (b) Door Open (feedback=10K) (c) Drawer Open (feedback=10K)



(d) Sweep Into (feedback=10K) (e) Window Open (feedback=10K) (f) Hammer (feedback=10K)

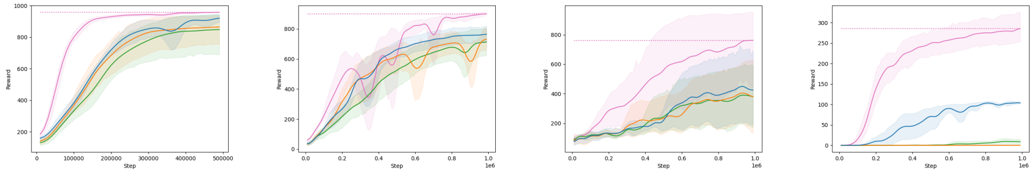
Figure 11: **Reward graphs in six manipulation tasks from Meta-World.** All methods are implemented based on MRN. Each subfigure describes the comparison results among three trajectory comparison methods: pairwise (green), sequential pairwise (orange), and root pairwise (blue). Pink lines describe the oracle performance using SAC with the true reward.

Task	# feedback	Oracle	Pairwise	Sequential Pairwise	Root Pairwise
Walker Walk	0.4K	957.3 ± 2.3	848.1 ± 170.5	864.4 ± 120.8	**919.8** ± 31.2
Cheetah Run	0.2K	886.9 ± 43.2	712.5 ± 117.0	730.0 ± 96.9	**763.5** ± 73.4
Quadruped Walk	1K	843.3 ± 247.3	380.7 ± 283.6	380.2 ± 319.6	**425.1** ± 327.3
Hopper Hop	4K	273.5 ± 47.0	8.9 ± 8.6	0.1 ± 0.1	**104.0** ± 3.5

Table 6: **Rewards after convergence in DMControl locomotion tasks using PEBBLE Baseline.** Two elements in each cell denote the average value and standard deviation of rewards across runs with 5 random seeds.

Task	# feedback	Oracle	Pairwise	Sequential Pairwise	Root Pairwise
Door Open	10K	100.0 ± 0.0	99.6 ± 0.9	99.8 ± 0.4	**100.0** ± 0.0
Drawer Open	20K	99.9 ± 0.3	82.2 ± 39.2	88.8 ± 25.0	**100.0** ± 0.0
Sweep Into	10K	88.8 ± 29.9	27.5 ± 47.8	41.2 ± 43.2	**71.8** ± 48.0
Window Open	1K	99.9 ± 0.3	49.4 ± 43.2	72.4 ± 39.0	**98.6** ± 3.1

Table 7: **Success rates in Meta-World manipulation tasks using PEBBLE Baseline.** Two elements in each cell denote the average value and standard deviation of success rates across runs with 5 random seeds.

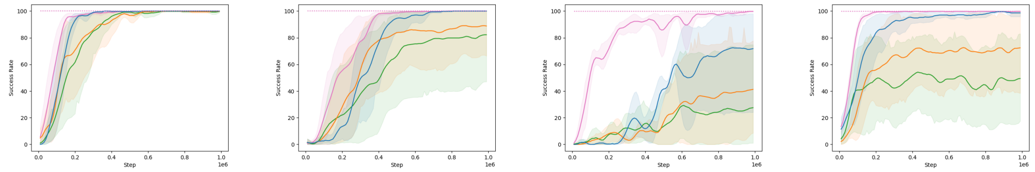


(a) Walker Walk (feedback=0.4K)

(b) Cheetah Run (feedback=0.2K)

(c) Quadruped Walk (feedback=1K)

(d) Hopper Hop (feedback=4K)



(e) Door Open (feedback=10K)

(f) Drawer Open (feedback=20K)

(g) Sweep Into (feedback=10K)

(h) Window Open (feedback=1K)

Figure 12: **Exp1: PEBBLE Baseline Experiment Results.** We compare three trajectory comparison methods: pairwise (green), sequential pairwise (orange), and root pairwise (blue). Pink lines describe the oracle performance using SAC with the true reward.

D.5 PEBBLE Baseline Experiments

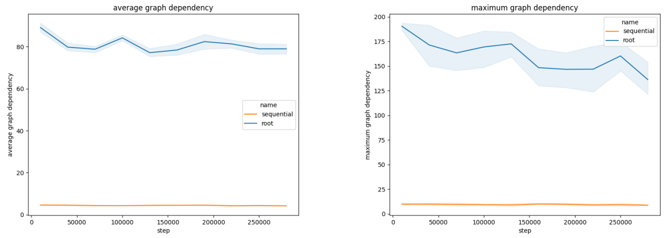
We train four tasks in DMControl (walker walk, cheetah run, quadruped walk, and hopper hop) and four tasks in Meta-World (door open, drawer open, sweep into, and window open), using 5 random seeds to check the validity of our method using PEBBLE baseline. The oracle performance is also evaluated from the same 5 random seeds. All tasks are trained for 1M steps from scratch.

Locomotion Tasks from DMControl. Results in Table 6 demonstrate that root and sequential pairwise comparison outperform baseline pairwise comparison by 14.0% and 1.3% on average, respectively. Especially, root pairwise comparison demonstrates its superiority over the baseline pairwise comparison for all four tasks: walker walk, cheetah run, quadruped walk, and hopper hop. Based on the standard deviation of rewards after convergence in walker walk and cheetah run tasks, root pairwise comparison shows the highest stability among the three trajectory comparison methods. In the hopper hop task, it is notable that only root pairwise comparison achieves an average reward larger than 100 while training the agent using pairwise or sequential pairwise comparison is unsuccessful.

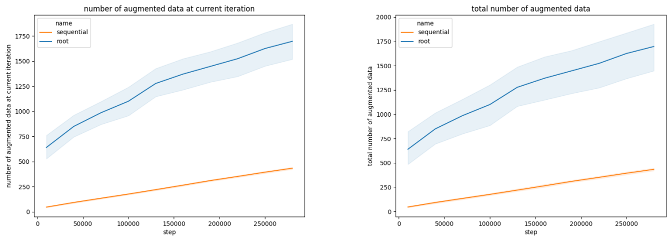
Manipulation Tasks from Meta-World. Results in Table 7 show that root and sequential pairwise comparison outperform baseline pairwise comparison by 43.2% and 16.8% on average, respectively.

Overall, the experiment results in Table 6 and Table 7 show that PEBBLE combined with SeqRank outperforms the PEBBLE baseline for eight robotic tasks. Performance graphs in Figure 12 show that root pairwise comparison enables faster convergence of the policy.

D.6 Data Dependency and Data Augmentation Measurements



(a) Average Graph Dependency (b) Maximum Graph Dependency



(c) Number of Augmented Preference Data at Current Iteration (d) Total Number of Augmented Preference Data

Figure 13: **Data dependency and the Number of Augmented Data while Training the Agent.** We compare sequential pairwise comparison (orange) and root pairwise comparison (blue).

We measure the number of augmented trajectory pairs and the graph dependency while training the DMControl Quadruped Walk task with 1000 feedback. We set $M = 100$ and the reward learning frequency as $1/30,000$ so that we sample 100 trajectories at each $30,000$ policy training steps to train the reward model. We use hyperparameters in Table 2. Figure 13 illustrates the average graph dependency, maximum graph dependency, number of augmented preference data at the current training step, and the cumulative number of augmented preference data.

Data Dependency. Theoretically, based on Lemma 3 and Lemma 4 on page 7 of the main paper, the upper bound of maximum graph dependency for sequential and root pairwise comparison is 10.11 and 195.97, respectively, when $N = 100$, $M = N + 1 = 101$, and $\delta = 0.03$. The experiment results in Figure 13 show that the maximum graph dependency for sequential and root pairwise comparison is 9.86 and 190.43, respectively. This shows that our theoretical analysis could be practically used to model the experimental results.

Number of Augmented Data. For the number of augmented data at the current training step, the range of the number of augmented data for sequential and root pairwise comparison is $[39.57, 46.00]$ and $[71.71, 640.29]$, respectively. The values are averaged across 10 random seeds. It is notable that the number of augmented data decreases as the training step increases for both sequential and root pairwise comparison. For root pairwise comparison, the number of augmented data at the first reward learning iteration, 640.29, is 7 times larger than the expected number of augmented data, 91.61. On the other hand, for sequential pairwise comparison, the number of augmented data at the first reward learning iteration, 46.00, is 1.19 times larger than the expected number of augmented data, 38.75. The results imply that both proposed methods, especially root pairwise comparison, perform better than the expected scenario and boost the early-stage reward learning process, while the data dependency does not explode and remains in a limited range.

D.7 Experiments with Real Human Feedback

For experiments with real human feedback, we show the following instruction before starting the experiment.

Instructions:

Welcome to the experiment! In this task, you will train a cheetah using three different methods to compare its running trajectories: pairwise comparison, sequential pairwise comparison, and root pairwise comparison. The goal is to make the cheetah run as fast as possible in the forward direction on the screen.

You will go through two runs, each consisting of 10 feedback sessions. In the first run, you will watch low-performing trajectories. In the second run, you will observe trajectories with varying performance.

During each feedback session, you will watch two videos of the cheetah running to the right side of the screen. Each video has 30 frames, and they will be played one after another. Your task is to select the preferred trajectory between the two shown on the screen within a time limit of 3 seconds.

We will use different methods for comparing trajectories:

Pairwise Comparison:
You will watch 20 different trajectories divided into 10 pairs (e.g., σ_1 and σ_{11}, σ_2 and σ_{12}).

Sequential Pairwise Comparison:
You will watch 11 different trajectories grouped into 10 pairs (e.g., σ_1 and σ_2, σ_2 and σ_3).

Root Pairwise Comparison:
Similar to sequential pairwise comparison, you will observe 11 different trajectories and compare each new trajectory with the previously most preferred one.

To indicate your preference during each query, simply click on the preferred trajectory within the given time. The chosen trajectory will be highlighted with a red box, and the next query will follow.

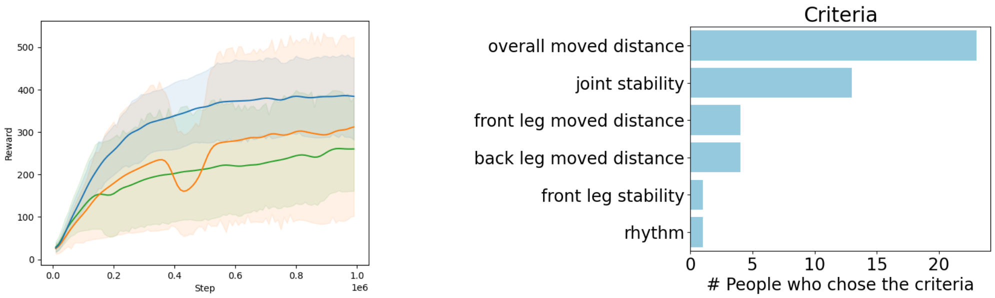
After the experiments end, the participants take a survey by answering the following questions.

Survey Questions:

Q1) Express the user satisfaction that you have experienced from each trajectory comparison method in levels from 1 to 5. A higher score indicates more satisfaction and less stress, while a lower score indicates less satisfaction and more stress. (1: strong stress, 2: weak stress, 3: no stress or satisfaction, 4: weak satisfaction, 5: strong satisfaction)

Q2) What were your own criteria for selecting one trajectory over the other? If you have multiple criteria, please write them down in order of priority.

Figure 14 (a) presents a histogram illustrating the user satisfaction scores for three trajectory comparison methods. The average scores resulted in 2.20, 3.00, and 3.93 for pairwise, sequential, and root pairwise comparison, respectively. In Figure 14 (b), the responses to Q2 are depicted, indicating that participants considered the overall moved distance of the agent as the most significant criterion to consider. Participants who responded for the moved distance of a specific leg prioritized the front leg over the back leg.



(a) Training Curves

(b) Preference Criteria

Figure 14: **Real Human Feedback Experiments in the Cheetah Run Task.** (a) We compare three trajectory comparison methods: pairwise (green), sequential pairwise (orange), and root pairwise (blue). The reward graphs are averaged across 5 human participants who trained the policy and the reward models from scratch. Each participant provides 200 feedbacks using each method. (b) The histogram shows the types of criteria and the number of people who chose them during the user study.

Experiment	# human feedback	# participants	Pairwise	Sequential Pairwise	Root Pairwise
Fine-tuning	20	30	860.10 ± 3.35	860.93 ± 2.28	**861.70** ± 2.25
Training from Scratch	200	5	260.2 ± 143.0	312.0 ± 276.5	**384.3** ± 124.6

Table 8: **Performance of Real Human Feedback Experiments in the Cheetah Run Task.** Two elements in each cell denote the average value and standard deviation of rewards across the participants.

We also perform statistical analysis using paired-sample t-test for (1) rewards and (2) user satisfaction scores. For both metrics, we use the following hypotheses.

- **Null Hypothesis (H0):** The mean difference between the methods (two chosen from pairwise, sequential pairwise, and root pairwise comparison) in terms of rewards after convergence (or user satisfaction scores) is equal to zero.
- **Alternative Hypothesis (Ha):** The mean difference between the methods (two chosen from pairwise, sequential pairwise, and root pairwise comparison, same as H0) in terms of rewards after convergence (or user satisfaction scores) is not equal to zero.

References

[1] Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. Generalization error bounds for classifiers trained with interdependent data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2005.

[2] Massih-Reza Amini and Nicolas Usunier. *Learning with partially labeled and interdependent data*. Springer, 2015.

[3] Rui-Ray Zhang and Massih-Reza Amini. Generalization bounds for learning under graph-dependence: A survey. *arXiv preprint arXiv:2203.13534*, 181:109272, 2022.

[4] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[5] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[6] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on robot learning (CoRL)*, 2020.

[7] Runze Liu, Fengshuo Bai, Yali Du, and Yaodong Yang. Meta-reward-net: Implicitly differentiable reward learning for preference-based reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[8] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[9] Hogun Kee, Minjae Kang, Dohyeong Kim, Jaegoo Choy, and Songhwai Oh. Sdf-based graph convolutional q-networks for rearrangement of multiple objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.