SAGE: State-Aware Grouping for Advantage Estimation in Long-Horizon Language Agent Reinforcement Learning

Anonymous ACL submission

Abstract

Group-based reinforcement learning (group-based RL) provides an effective path for post-training of Large Language Models (LLMs). However, step-level credit assignment remains difficult in long-horizon text interaction tasks. Existing methods mostly rely on discrete state matching or historical context consistency to build local groups. They often face a trade-off between overly coarse grouping and poor sample utilization. To address this issue, we propose State-Aware Grouping for Advantage Estimation (SAGE). SAGE builds a dual-branch state representation. This comprises the current state and local variation. It captures both the state semantics and changes relative to recent history. We also introduce a pre-bucketing mechanism based on relative progress to reduce mixing of steps across stages. Then, we perform local greedy clustering within each bucket to form state clusters. We define the training signal as a weighted sum of relative advantage within the same input sample group and relative advantage within state cluster. This design avoids reliance on discrete exact matching. Instead, it builds local groups based on state semantics, historical changes, and relative stages. We evaluate SAGE on ALFWorld and WebShop. Results show SAGE achieves the best performance on ALFWorld success rates and the WebShop task success rate, while remaining competitive on the WebShop task score. Further analysis shows the dual-branch state representation, relative progress bucketing, and local greedy clustering play important roles in improving performance.

1 Introduction

Large Language Models (LLMs)(Yang et al., 2025; DeepSeek-AI et al., 2025; Team et al., 2025a; OpenAI, 2023; Dubey et al., 2024; Zeng et al., 2024; Team et al., 2025b) are evolving from single-turn text generation models into language agents. These agents can continuously perceive, reason, and act in external environments. Unlike regular text generation tasks, long-horizon interaction tasks involve multi-step decisions, sparse rewards, and delayed feedback. The choices a model makes in early steps can affect long subsequent interactions. Therefore, providing effective credit-assignment signals for intermediate decisions is a key issue in improving training of long-horizon agents.

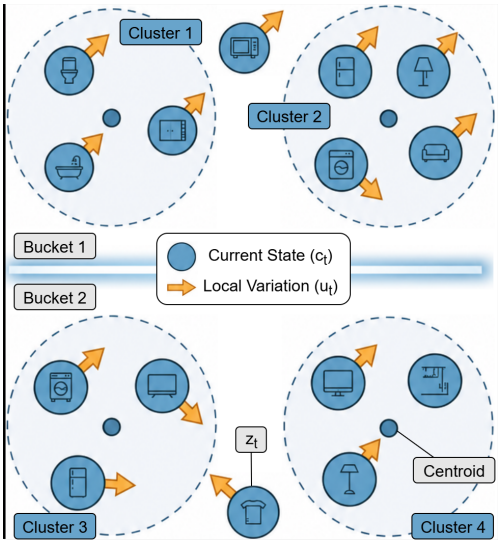

Figure 1: State-aware clustering: SAGE buckets steps by progress and clusters them by state & local variation.

Reinforcement learning (RL)(DeepSeek-AI et al., 2025; Kool et al., 2019; Ahmadian et al., 2024; Yu et al., 2026; Zheng et al., 2025) is an important post-training method for enhancing an agent's decision-making ability. Group-based RL(DeepSeek-AI et al., 2025; Feng et al., 2026; He et al., 2026) does not rely on an extra value network. Thus, it offers good training efficiency and scalability, making it a major direction in large model post-training. However, most existing group-based RL methods are better suited for single-turn generation or trajectory-level optimization. When applied directly to long-horizon agent tasks, they often struggle to provide effective credit assignment

signals for intermediate steps. Recent step-level methods have started building local relative advantage groups at intermediate steps. For example, GiGPO(Feng et al., 2026) builds step-level groups via anchor state matching. HGPO(He et al., 2026) further uses historical context consistency to build hierarchical groups. This improves local advantage estimation in long-horizon tasks. However, these methods still largely rely on discrete state matching or strict historical consistency. They easily face a contradiction between grouping granularity and sample utilization. If the grouping is too coarse, states that look similar on the surface but belong to different decision stages might merge. If the grouping is too fine, the number of samples available for comparison decreases, leading to unstable advantage estimation. Therefore, long-horizon agent training needs a more flexible local grouping method. This approach ensures that comparison targets share a similar decision context while maintaining enough sample utilization.

As shown in Fig. 1, we propose State-Aware Grouping for Advantage Estimation (SAGE). SAGE's core idea is that step-level relative advantage estimation should no longer rely merely on whether discrete states match exactly. Instead, it builds local relative advantage groups that better fit the decision context, based on state semantics, recent changes, and task progress. In this way, the training signal for each step can use both the overall contrast within the same input sample and the local relative advantage under similar state conditions. This provides more fine-grained supervision for intermediate decisions in long-horizon interactions.

We evaluate SAGE on ALFWorld(Shridhar et al., 2021) and WebShop(Yao et al., preprint). Experimental results show that SAGE gets the best results on ALFWorld's in-distribution and out-of-distribution success rates (In-Success & Out-Success), as well as the WebShop task success rate. It also remains competitive on the WebShop task score. Further analysis shows that state-aware local grouping helps mitigate the issues of state over-merging and low sample utilization found in discrete matching, thereby improving the completion rate of long-horizon tasks.

Our main contributions are as follows:

- We point out existing step-level group-based RL methods rely on discrete matching or historical consistency. They struggle to balance state conditionality and sample utilization.
- We propose SAGE. It builds step-level relative advantage groups fitting long-horizon context through state-aware grouping.
- We combine the sample-level relative advantage with the state-level local relative advantage. This improves step-level credit assignment in long-horizon interaction tasks.
- We validate the effectiveness of SAGE on ALFWorld & WebShop. Results show SAGE improves task success rate, while remaining competitive on WebShop continuous score.

2 Related Work

LLMs have evolved from single-turn text generation models into language agents(Gao et al., 2025). These agents can continuously perceive, reason, and act in environments. To improve their decision-making ability in complex tasks, Reinforcement Learning (RL) has become an important post-training method. Early methods often used classic policy optimization algorithms like PPO(Schulman et al., 2017). However, they usually rely on extra value networks, leading to high training and tuning costs. Recently, group-based RL methods like RLOO(Ahmadian et al., 2024) and GRPO(DeepSeek-AI et al., 2025) build relative advantages directly through intra-group comparisons on the same input. They balance training stability and computational efficiency without needing an extra critic. Meanwhile, multi-agent reinforcement learning methods, such as REMA(Wan et al., 2026), further improve policy optimization through multi-agent collaboration and interactive feedback. Together, they form an important technical path in current language model post-training. However, these methods were mostly designed for single-turn generation or coarse-grained optimization. When applied directly to long-horizon interaction tasks, they often struggle to provide effective supervision signals for intermediate decisions.

Focusing on this issue, recent studies have started to look at step-level credit assignment in long-horizon agents. Compared to optimizing the whole trajectory as a single unit, step-wise methods can directly optimize intermediate steps. This makes them better suited for handling sparse rewards and delayed feedback in long-horizon tasks. Representative methods like GiGPO(Feng et al., 2026) build step-level groups via anchor state matching. HGPO(He et al., 2026) further uses

historical context consistency to build hierarchical groups improving local advantage estimation. These methods prove importance of step-level local relative advantage for long-horizon training. However, their grouping methods still rely on discrete state matching or strict historical consistency. They often face a trade-off between overly coarse grouping and poor sample utilization. Different from approaches, SAGE builds local relative advantage groups based on continuous state representations. It also introduces recent historical changes and relative progress constraints. This forms step-level groups better matching state semantics and provides more stable advantage estimation.

3 Method

3.1 Task Setup and Overall Framework

We consider long-horizon text interaction tasks. Given an input sample x, the current policy π_θ samples M interaction trajectories $\{\tau^{(m)}\}_{m=1}^{M}$ under the same task conditions. Let $p_t^{(m)}$, $a_t^{(m)}$, $r_t^{(m)}$, and $o_t^{(m)}$ denote the policy context, action, immediate reward, and anchor observation for state encoding at step t of the m-th trajectory. Here, T_m is the effective length of the trajectory. Thus, the m-th trajectory is written as:

$$\tau^{(m)} = \{(p_t^{(m)}, a_t^{(m)}, r_t^{(m)}, o_t^{(m)})\}_{t=0}^{T_m-1} \tag{1}$$

For the same input sample x, we denote index set of sampled steps as $\mathcal{U}(x) = \{(m,t) \mid 0 \le t < T_m\}$.

SAGE builds the training signal at the step level. For each $(m,t) \in \mathcal{U}(x)$, our method first builds a continuous state representation from $o_t^{(m)}$. Then, it divides the steps into different buckets based on their relative progress within the same input sample. Next, it performs local greedy clustering inside each bucket to find the local state cluster $C(m,t)$ that the step belongs to. Based on this, SAGE computes two relative advantages: one within the same input sample group, and another within the state cluster. It computes their weighted sum to get the final advantage $A_t^{(m)}$ for that step. This advantage is then used to update the policy for the response at that step. The following sections provide the specific definitions of state representation, local grouping, and advantage estimation. Figure 2 gives an overview of the SAGE pipeline.

3.2 Step State Representation

For simplicity, we omit trajectory superscript (m) in this section. Given the anchor observation text o_t at step t, let $\{h_{t,\ell}\}_{\ell=1}^{L_t}$ be last-layer token representations from the model. Let $\{m_{t,\ell}\}_{\ell=1}^{L_t}$ be the corresponding valid token masks. We get the sentence-level representation via masked mean pooling:

$$e_t = \frac{\sum_{\ell=1}^{L_t} m_{t,\ell} h_{t,\ell}}{\sum_{\ell=1}^{L_t} m_{t,\ell}} \tag{2}$$

Let $\mathrm{Norm}(v)$ denote the safe normalization operator. It applies ℓ_2 normalization to non-zero vectors and keeps zero vectors as zero. Thus, we write the current state representation as $c_t = \mathrm{Norm}(e_t)$.

Next, we construct the historical context using a short history window of length K. Let $\eta \in [0,1)$ be the decay coefficient. We define the historical representation of step t as:

$$\bar{c}_t = \mathrm{Norm}\left(\sum_{k=1}^{\min(K,t)} (1-\eta)\eta^{k-1} c_{t-k}\right), t \ge 1. \tag{3}$$

Based on this, we use the difference between the current state and the history to construct the local variation representation. We denote this as $u_t = \mathrm{Norm}(c_t - \bar{c}_t)$. For the first step, since there is no valid history, we set $u_0 = \mathbf{0}$.

Finally, we combine current state and local variation branches into unified step representation:

$$z_t = \mathrm{Norm}([\sqrt{\omega_{\mathrm{cur}}}\, c_t; \sqrt{\omega_{\mathrm{inn}}}\, u_t]) \tag{4}$$

where $\omega_{\mathrm{cur}}, \omega_{\mathrm{inn}} \ge 0$ and $\omega_{\mathrm{cur}} + \omega_{\mathrm{inn}} = 1$, and $[\cdot;\cdot]$ denotes vector concatenation. Here, z_t summarizes dual-branch state encoding of SAGE. z_t is a conceptual concatenated representation, while clustering uses the equivalent branch-wise score. In subsequent local grouping, we do not directly calculate $\cos(z_i, z_j)$. Instead, we use a branched matching score for this representation. The subscript cur indicates current state branch, and inn indicates local variation branch (innovation branch).

3.3 Local Greedy Clustering under Relative Progress Constraints

After getting the step representations, SAGE builds local state clusters within the same input sample. For step t of trajectory $\tau^{(m)}$, we first assign it to a bucket based on its relative progress:

$$b_t^{(m)} = \min\left(\left\lfloor \frac{tB}{T_m} \right\rfloor,\ B-1\right) \tag{5}$$

where B is the number of buckets, and T_m is the effective length of the trajectory. Thus, we define

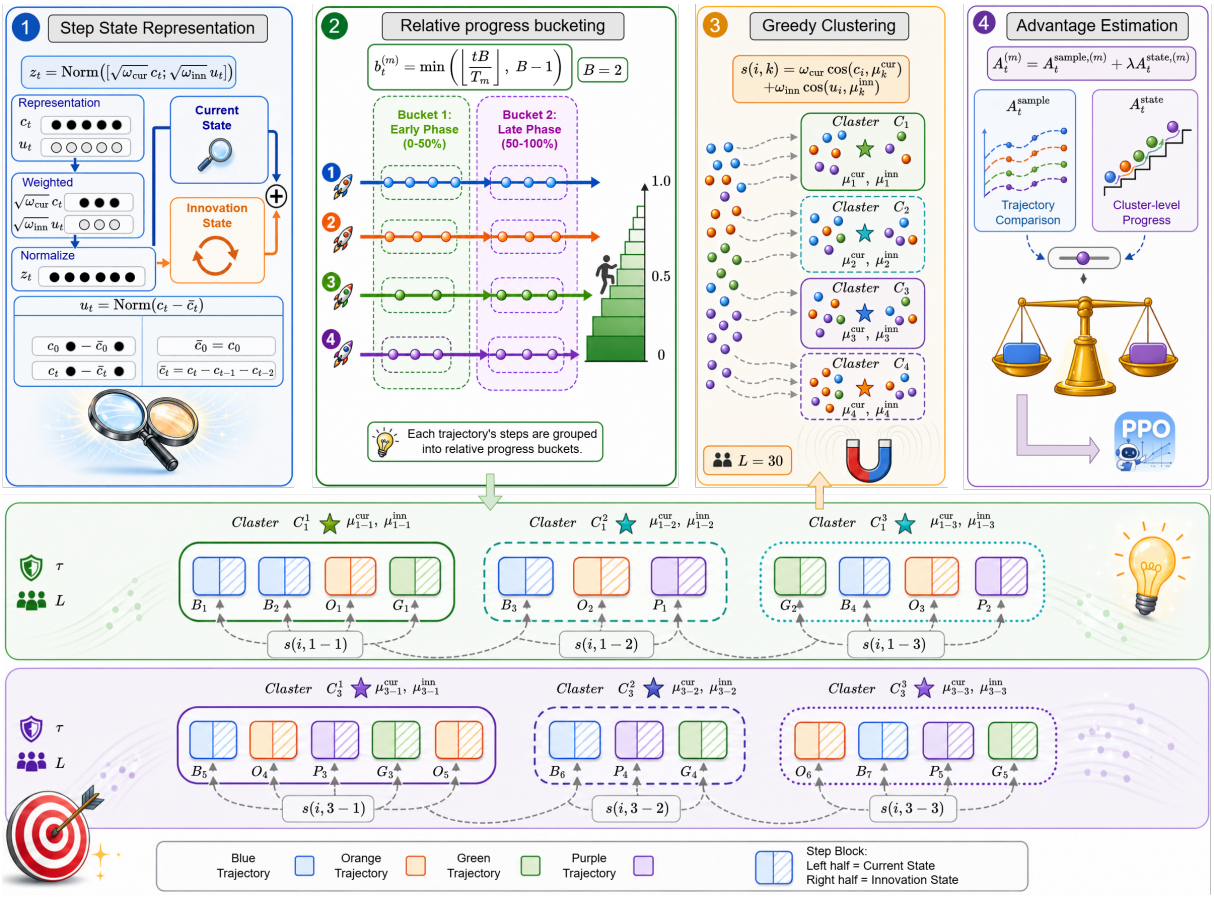

Figure 2: Overview of SAGE. The top panel shows the full pipeline for step-level advantage estimation, including two-branch state representation, relative-progress bucketing, greedy clustering, and advantage computation. The bottom panel illustrates the bucket-wise clustering process and cluster updates.

the b-th bucket under the same input sample x as:

$$\mathcal{P}(x, b) = \{(m, t) \in \mathcal{U}(x) \mid b_t^{(m)} = b\} \quad (6)$$

We perform clustering independently inside each bucket. We sequentially assign steps to clusters.

We use local greedy clustering. Suppose the first member j_k initializes a cluster C_k. We initialize the cluster's current state branch representation and local variation branch representation as $\mu_k^{\text{cur}} = c_{j_k}$ and $\mu_k^{\text{inn}} = u_{j_k}$, respectively. In our default implementation, cluster representation uses a centroid update. When a new step is added to the cluster, we update the cluster center to normalized mean direction of corresponding branch representations within the cluster. For a step i to be assigned, its matching score with cluster C_k is defined as:

$$s(i, k) = \omega_{\text{cur}} \cos(c_i, \mu_k^{\text{cur}}) + \omega_{\text{inn}} \cos(u_i, \mu_k^{\text{inn}}),$$
$$\|u_i\|_2 > 0 \text{ and } \|\mu_k^{\text{inn}}\|_2 > 0. \quad (7)$$

If either innovation vector is zero, we use only the current-state similarity.

Let L be the maximum capacity of a single cluster, and let τ be the matching threshold. We denote the candidate cluster set for step i as $\mathcal{K}_i = \{k \mid |C_k| < L\}$. If $\mathcal{K}_i$ is not empty, we select:

$$k^\star = \arg\max_{k \in \mathcal{K}_i} s(i, k) \quad (8)$$

If $s(i, k^\star) \geq \tau$, we assign step i to cluster $C_{k^\star}$. We also update cluster representation according to centroid update strategy. Otherwise, we create a new state cluster with i as its first member. If $\mathcal{K}_i = \varnothing$, we directly create a new state cluster for this step as well. After fixing processing order inside a bucket, we can determine the local state cluster assignment for each step. The Appendix provides stable sorting rules and other implementation details we use.

3.4 Two-Level Relative Advantage Estimation and Policy Update

After local grouping is complete, SAGE builds two relative advantages for each step. We again omit un-

necessary superscripts when the meaning is clear. Assume the response at step t has L_t^{resp} tokens. Let $\rho_{t,1}^{(m)}, \ldots, \rho_{t,L_t^{\text{resp}}}^{(m)}$ be their token-level rewards. We denote the scalar response score of this step as $q_t^{(m)} = \sum_{\ell=1}^{L_t^{\text{resp}}} \rho_{t,\ell}^{(m)}$. In our implementation, this is the token-level reward sum used by PPO. For a group $\mathcal{G}$, let $\mathrm{RelNorm}_{\mathcal{G}}(\cdot)$ denote relative normalization: mean normalization subtracts the group mean, while mean-std normalization further divides by the group standard deviation. Following GiGPO, we use mean-std normalization on ALFWorld and mean normalization on WebShop. For the same input sample x, we define the relative advantage within the input sample group as:

$$A_t^{\text{sample},(m)} = \mathrm{RelNorm}_{\mathcal{U}(x)}\left(q_t^{(m)}\right). \quad (9)$$

Next, for step t of trajectory $\tau^{(m)}$, we define its discounted step return as:

$$G_t^{(m)} = \sum_{k=t}^{T_m-1} \gamma^{k-t} r_k^{(m)} \quad (10)$$

where $\gamma \in (0, 1]$ is discount factor. Let $C(m, t)$ be local state cluster containing this step. We then write relative advantage within state cluster as:

$$A_t^{\text{state},(m)} = \mathrm{RelNorm}_{C(m,t)}\left(G_t^{(m)}\right). \quad (11)$$

Finally, we define overall advantage of step t as:

$$A_t^{(m)} = A_t^{\text{sample},(m)} + \lambda A_t^{\text{state},(m)} \quad (12)$$

where $\lambda \geq 0$ is the weight of the local state term. Thus, SAGE follows GiGPO's two-level advantage structure while replacing exact state matching with state-aware local grouping. After getting $A_t^{(m)}$, we use it as the advantage for all valid response tokens of that step in the subsequent PPO update.

4 Experiments

4.1 Experimental Setup

We evaluate SAGE on two long-horizon agent benchmarks: ALFWorld and WebShop. The former tests multi-step planning and manipulation skills in text environments. The latter tests goal-oriented web interaction skills. During evaluation, ALFWorld reports In-Success & Out-Success rates. WebShop reports the task score and task success rate.

We compare SAGE with three categories of methods. The first category is prompting methods. This includes Closed-source models like GPT-4o and Gemini-2.5-Pro. It also includes vanilla prompting, ReAct, and Reflexion based on Qwen2.5-1.5B-Instruct. The second category is general reinforcement learning baselines. This includes PPO, RLOO, and GRPO. The third category contains relevant step-level group-based strong baselines, including GiGPO and HGPO. To ensure fair comparison, all RL methods share same backbone, rollout budget, and training epochs.

For our implementation, all RL training methods are based on Qwen2.5-1.5B-Instruct. Unless stated otherwise, we use a rollout group size of 8 and a training batch size of 16. The maximum prompt length is 4096, and maximum response length is 512. We set learning rate to 1.25×10^{-6} and the warmup ratio to 0.02. For SAGE, we use contextual hidden states to build step state representations. We set the history window size to $K = 2$. We set the weights of current state branch and local variation branch to 0.7 and 0.3, respectively. We enable relative progress bucketing by default before local grouping, setting bucket count to $B = 2$. The Appendix provides other implementation details.

4.2 Main Experimental Results

Table 1 reports the main experimental results of SAGE and various baselines on ALFWorld and WebShop. Overall, SAGE achieves the best results on three main success rate metrics. The ALFWorld In-Success rate is 96.1±0.8. The out-of-distribution success rate is 92.87±0.49. The WebShop task success rate is 79.1±0.6.

Looking at method categories, prompting methods generally perform weakly on long-horizon interaction tasks. In contrast, RL training methods bring clear improvements. This shows relying on prompt design struggles handling intermediate decisions and delayed feedback in long-horizon tasks. Policy optimization remains important way to improve behavioral abilities of language agents.

Within RL methods, step-level group-based methods generally outperform general RL baselines. This indicates step-level local relative advantage play an important role in long-horizon credit assignment. Compared to GiGPO and HGPO, SAGE further introduces continuous state representations, local variation information, and relative progress constraints. As a result, it builds local relative advantage groups better matching state se-

Model	Type	Method	ALFWorld In-Success	ALFWorld Out-Success	WebShop Success Rate	WebShop Task Score
Closed source	Prompting	GPT-4o	48.0	46.0	23.7	31.8
		Gemini-2.5-Pro	60.3	50.5	35.9	42.5
Qwen2.5-1.5B-Instruct	Prompting	Qwen2.5	4.1	-	5.2	23.1
		ReAct	12.8	-	11.3	40.1
		Reflexion	21.8	-	21.9	55.8
	RL Training	PPO	54.4±3.1	-	51.5±2.9	73.8±3.0
		RLOO	69.7±2.5	68.7±10.7	52.1±6.7	73.9±5.6
		GRPO	72.8±3.6	70.1±2.5	56.8±3.8	75.8±3.5
		GiGPO (K=2)	90.16±0.22	84.76±2.83	66.53±1.92	84.95±2.57
		HGPO (K=2)	92.77±1.08	90.16±0.78	71.54±4.00	85.56±2.86
		GiGPO (K=4)	93.29±1.07	91.53±1.99	73.24±2.25	86.80±1.60
		HGPO (K=4)	94.85±0.92	92.12±1.63	78.12±2.06	**90.64±1.05**
		SAGE (Ours)	**96.1±0.8**	**92.87±0.49**	**79.1±0.6**	86.35±0.25

Table 1: Main experimental results on ALFWorld and WebShop.

mantics. Compared with strongest success rate baseline HGPO ($K = 4$), SAGE achieves further improvements on ALFWorld in/out and WebShop success rate. This indicates that state-aware grouping can more stably improve task completion rate. However, SAGE does not beat HGPO ($K = 4$) on WebShop Task Score. It is also slightly lower than GiGPO ($K = 4$). This suggests SAGE more directly optimizes whether final task is completed. The continuous WebShop score might also be affected by factors like fine-grained product matching, path selection, and local operation quality.

4.3 Ablation Study

To analyze role of SAGE's components and its key implementation choices, we run ablation and configuration variant experiments on WebShop using Qwen2.5-1.5B-Instruct. Except for explicitly modified parameters, all variants keep the same settings as full SAGE. Table 2 reports the main results. The full SAGE gets highest result on the WebShop Task Success Rate at 79.1±0.6.

These ablation items correspond to replacing or removing different designs in SAGE. `Sample-only` removes the state cluster relative advantage, keeping only the global relative comparison within the input sample group. `Current-only` uses only the current observation representation, removing the local variation branch. `EMA State Feature` replaces the default local variation with an exponential moving average of historical states. `Token Embedding` replaces the contextualized hidden states with pooled token embeddings. `Weighted-Sum Fusion` computes a weighted sum of the current state branch and the local variation branch before normalization. `Current Zero-History` changes how steps with no history are handled by setting the history representation to the current state. `Mean-Std Norm` changes WebShop from mean normalization to mean-std normalization. `First-Fit Assignment` alters the nearest cluster assignment by picking the first available cluster in order. `First-Sample Center` fixes the cluster center to the first sample instead of using a centroid update.

Variant	WebShop Success Rate	WebShop Task Score
Sample-only	56.8±3.8	75.8±3.5
Current-only	41.0±1.2	55.2±0.7
EMA State Feature	77.35±1.95	87.25±0.15
Token Embedding	59.47±0.88	73.72±2.29
Weighted-Sum Fusion	61.5±0.2	79.05±2.05
Current Zero-History	73.25±1.35	85.9±0.2
Mean-Std Norm	70.5±1.0	83.25±0.05
First-Fit Assignment	72.85±1.35	83.9±1.4
First-Sample Center	77.93±0.2	**89.43±0.12**
SAGE	**79.1±0.6**	86.35±0.25

Table 2: Ablation results of major components and implementation choices on WebShop.

The results show that the state cluster relative advantage is a key part of SAGE. The success rate of `Sample-only` is much lower than the full SAGE. This indicates that relying solely on global comparisons inside the input sample group fails to provide stable step-level credit assignment sig-

Similarity thresh τ	WebShop Task Success Rate	WebShop Task Score
0.88	69.14±2.74	80.97±1.17
0.90	**79.1±0.6**	**86.35±0.25**
0.92	64.65±0.55	70.15±0.75
0.94	76.2±0.4	84±1.2
0.96	75±0.8	86.3±1.1
0.98	71.1±1.6	85.15±0.15

Table 3: Sensitivity to similarity threshold τ. τ controls the minimum matching similarity for a step to join an existing state cluster.

Current/ Innovation	WebShop Task Success Rate	WebShop Task Score
1.0/0.0	41±1.2	55.2±0.7
0.8/0.2	77.5±0.2	**87.55±1.05**
0.7/0.3	**79.1±0.6**	86.35±0.25
0.6/0.4	77.15±0.15	86.75±0.55
0.3/0.7	73.45±1.95	82.4±1.2

Table 4: Sensitivity to state branch weights. Current/Innovation denotes the relative weights assigned to the current observation semantic similarity branch and the local variation branch, respectively.

Factor	Setting	WebShop Task Success Rate	WebShop Task Score
Progress Bucket	$B = 0$	54.9±0.6	71.6±0
	$B = 2$	**79.1±0.6**	86.35±0.25
	$B = 4$	69.3±0.2	77.8±1
	$B = 6$	75.2±0.6	**86.45±0.85**
History Window	$K = 0$	41±1.2	55.2±0.7
	$K = 2$	**79.1±0.6**	**86.35±0.25**
	$K = 4$	74±0.2	85.1±0.1
	$K = 6$	67.2±0.4	81.2±1.7
Cluster Capacity	$L = 15$	75±2	**88.15±1.55**
	$L = 30$	**79.1±0.6**	86.35±0.25
	$L = 60$	75.57±1.77	84.61±0.21
	$L =$ None	72.5±0.2	81.05±1.05

Table 5: Sensitivity to progress bucketing, history window, and cluster capacity.

nals. State representation ablations also show that current state semantics and local variation information must work together. `Current-only` suffers the most obvious performance drop. Also, using contextualized hidden states is clearly better than simple token embedding pooling. This shows that effective local grouping relies not only on the static semantics of the observation but also on its changes relative to recent history.

In addition, branch fusion, advantage normalization, and clustering strategies also affect training performance. The drop in `Weighted-Sum Fusion` shows that mixing the two state branches too early may weaken their complementarity. The results of `Mean-Std Norm` show that extra standard deviation scaling does not necessarily yield more stable advantage estimates. `First-Fit Assignment` falls below the default setting, showing that selecting the most similar cluster among candidates suits SAGE's local grouping process better. Notably, `First-Sample Center` gets a higher task score than the full SAGE, but its success rate is slightly lower. This indicates that the centroid update favors the task completion rate, while fixing the center to the first sample may align more with WebShop's continuous scoring items.

4.4 Hyperparameter Analysis

We examine the impact of the similarity threshold τ on local grouping. Except for the explicitly modified parameters, all other settings match the full SAGE in Table 2. τ controls the minimum matching similarity needed for a new step to join an existing cluster. The full SAGE uses a default of $\tau = 0.90$.

Table 3 shows that the similarity threshold clearly affects the quality of SAGE's local grouping. The default setting $\tau = 0.90$ gets both the highest task success rate and the highest task score. This indicates that this threshold strikes a good balance between state differentiation and sample utilization. A lower threshold merges different states more easily. This weakens state-conditioned relative comparisons. A higher threshold creates finer cluster divisions. This might reduce the stability of the relative advantage estimation inside the cluster. Overall, SAGE must avoid grouping that is either too coarse or too fine. A moderate similarity threshold is important for ensuring training performance.

We further examine weights of current state branch and local variation branch. The full SAGE uses a default weight configuration of 0.7/0.3. Table 4 shows that the local variation branch is crucial. When we completely remove this branch (1.0/0.0), performance drops clearly. After adding it, the success rate rises significantly, and the default configuration achieves the highest success rate. This indicates that recent state changes provide key dynamic information. This helps distinguish states that look similar on the surface but have different decision stages or transition trends. At the same time, the 0.8/0.2 setting gets a higher Task Score but lower success rate. This shows that the branch ratio affects model's preference for completion rate versus continuous scoring. Overall, a combination dominated by current state semantics and supplemented by local variation is more stable.

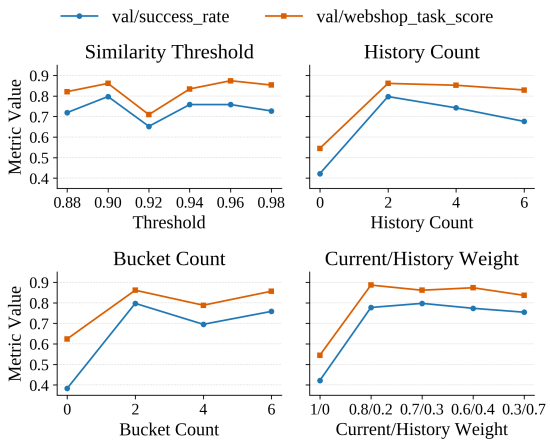

Figure 3: Validation performance under four hyperparameter studies.

We further examine the number of relative progress buckets B, the history window size K, and the cluster capacity upper limit L. The full SAGE defaults to $B = 2$, $K = 2$, and $L = 30$.

Table 5 shows that progress constraints, historical modeling, and cluster capacity all affect the quality of local state clusters. Performance drops clearly when we disable progress bucketing. This shows that mixing steps across stages weakens the effectiveness of local relative advantage. The default $B = 2$ achieves the highest task success rate. This indicates that moderate stage division helps reduce cross-stage mixing. However, finer division does not always bring further gains. Over-division might instead lower sample utilization within the bucket. The history window experiments further confirm the importance of local variation modeling. When $K = 0$, performance is noticeably lower than the full SAGE. The default $K = 2$ works best. This shows that a short history can already provide effective dynamic information. If we expand the window further, performance actually drops. A longer history might introduce noise that is weakly related to the current decision. The cluster capacity upper limit mainly affects the granularity of local grouping. The default $L = 30$ achieves the highest success rate. This suggests that properly limiting cluster size helps avoid over-merging. A smaller capacity ($L = 15$) yields a higher task score but a slightly lower success rate. This reflects a trade-off between the two metrics.

Combining Tables 3, 4, and 5, SAGE's hyperparameters influence training performance mainly by tuning the granularity and historical conditionality of local state clusters. Overly coarse grouping easily mixes states from different stages. Conversely, overly fine grouping might weaken the stability of advantage estimation inside the cluster.

Besides the final task metrics in the tables, Figure 3 plots performance trend curves on WebShop. These are based on validation records from four hyperparameter experiments. They examine the impact of the similarity threshold, state branch weights, history window size, and progress bucket count on model performance. Looking at the trends, the metrics do not change monotonically with grouping granularity. Instead, they show more stable validation performance around medium settings. Overly coarse divisions merge different states. This weakens the state-conditioned advantage. Overly fine divisions cause a lack of samples within clusters. This makes advantage estimates and intra-group statistics more unstable. Furthermore, we gathered statistics on the distribution of state clusters formed under different settings. We place the related bar charts in the Appendix due to their large number. The clustering statistics show that better-performing settings usually correspond to a balanced cluster size distribution. This avoids the high variance caused by massive single-sample clusters. Overall, these results show that SAGE's effectiveness comes not only from finer state division. Most importantly, it relies on maintaining a moderate grouping granularity that balances state conditionality and intra-cluster statistical stability.

5 Conclusion

We propose State-Aware Grouping for Advantage Estimation (SAGE) for step-level credit assignment in long-horizon text interaction tasks. Our method first builds a continuous state representation from anchor observation. This representation consists of a current state branch and a local variation branch. Next, it performs local greedy grouping under relative progress constraints. Finally, we define training signal for each step as a weighted sum of two relative advantages: one within the same input sample group, and the other within the state cluster. Experimental results on ALFWorld and WebShop show SAGE stably improves the main success rate metrics. It also remains competitive on the WebShop task score. In summary, building step-level local relative advantage groups based on state representations, stage constraints, and local conditional comparisons is an effective way to improve credit assignment in long-horizon interaction scenarios.

Limitations

Our method still has several limitations. First, due to limited computing resources, we have not fully verified our method on larger-scale models. Therefore, SAGE's benefits, stability, and scaling laws on stronger foundation models need further study. Second, we use hard bucketing, hard grouping, and a local greedy assignment strategy with a fixed threshold. This implementation is fairly simple. However, it might limit the stability of relative advantage estimation on boundary states and small-sample clusters. Finally, our method clusters states based on vector representations. This naturally has the potential to extend to multimodal state representations. But our current experiments mainly focus on long-horizon tasks in text environments. We have not yet verified its applicability on multimodal benchmarks.

Future work can proceed in three directions. First, we plan to systematically evaluate SAGE's effectiveness and scalability on larger models. Second, we will explore more flexible stage division and local grouping mechanisms. This will improve the robustness of local relative advantage. Third, we aim to extend SAGE to multimodal long-horizon interaction scenarios. This will verify its applicability in joint decision-making tasks involving vision, text, and other modalities.

Acknowledgments

Acknowledgments omitted for anonymous review.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, and et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, , and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2026. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 1.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. 2026. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. *arXiv preprint arXiv:2602.22817*.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free!

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, and et al. 2025a. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, and et al. 2025b. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599.

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and 1 others. 2026. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 38:126621–126667.

An Yang, Anfeng Li, Baosong Yang, and et al. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. preprint. Webshop: Towards scalable real-world web interaction with grounded language agents. In *ArXiv*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, and et al. 2024. Chatglm: A family of large language models from GLM-130B to GLM-4 all tools. *CoRR*, abs/2406.12793.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

A Appendix

A.1 Additional details

To match the descriptions in the main text, this appendix provides three additional implementation details for the main experiments.

First, the main text uses r_t to denote the reward during environment interaction. In our main experiments, we also apply an invalid-action penalty with a coefficient of 0.1. This penalty is subtracted from the token-level reward score and from the scalar step-return value used before advantage normalization.

Second, the main text introduces a unified representation z_t to summarize SAGE's dual-branch state encoding. During actual local grouping, we do not directly compute $\cos(z_i, z_j)$. Instead, we use a branched matching score based on this representation. If the local variation branches of both the target step and the candidate cluster are valid, the matching score combines the weighted similarities of the current state branch and the local variation branch. If the local variation branch is missing on either side, the matching score drops back to the similarity based solely on the current state branch. In our experiments, we set ω_{cur} and ω_{inn} according to the formulas in the main text, and their sum equals 1.

Third, the main text denotes the capacity upper limit of a single local state cluster as L and the matching threshold as τ. During local greedy clustering, we first calculate matching scores only among candidate clusters below the capacity limit and select the best one. If the score of this optimal candidate cluster is at least τ, we add the current step to it. Otherwise, we create a new state cluster for it. If all existing clusters reach the capacity limit, we also directly create a new state cluster. By default, SAGE uses `nearest_center` as the assignment rule and uses `centroid` to update the cluster center. This means that after adding a sample, we update the cluster representation to the normalized mean direction of the member representations. In the ablation study, we also examine the `first_fit` assignment rule and the `first_sample` cluster center strategy. In our implementation, we stably sort the samples inside a bucket by step index, trajectory identifier, and original order. We then complete the assignment process in this precise order.

We use Qwen2.5-1.5B-Instruct, ALFWorld, and WebShop only for research training and evaluation purposes. Their use is consistent with their intended role as publicly available models and benchmarks for studying language agents and reinforcement learning methods.

SAGE follows GiGPO's task-specific advantage normalization: ALFWorld uses mean-std normalization, while WebShop uses mean normalization. The anchor observation uses the model's contextualized hidden states to build a sentence-level representation. The step state representation is formed by two branches: the current state branch and the local variation branch. We set the history window size to 2 and the history decay coefficient to 0.95. The two branches remain relatively independent via concatenation. During cluster matching, the weights of the current state branch and the local variation branch are 0.7 and 0.3, respectively. For steps lacking historical states, history-missing steps use a zero innovation vector, not "zero vector for the history branch. Before local grouping, we enable relative progress bucketing and default to 2 buckets. During in-bucket assignment, we select the candidate cluster with the highest similarity that has not reached its capacity limit. As new members join, we update the cluster center to the normalized mean direction of the representations in the cluster. The capacity limit for a single state cluster is 30. We set the similarity threshold by task: ALFWorld uses $\tau = 0.80$, while WebShop uses $\tau = 0.90$. Except for this threshold, the default settings for the full SAGE are identical for both tasks.

Due to the limited training hardware available for this study, we did not perform reinforcement learning training on Qwen2.5-7B-Instruct. Therefore, all open-source model training and ablation experiments in this paper are based on Qwen2.5-1.5B-Instruct.

A.2 Computational Resources

All experiments were conducted on 8 NVIDIA A100 40GB GPUs. The models were trained for approximately 250 epochs, with a total training time of around 23 hours. The experiments use Qwen2.5-1.5B-Instruct as the backbone model.

A.3 Analysis of State Cluster Size Distribution

Figures 4, 6, 5, and 7 show the step-level group size distribution produced by SAGE's local greedy clus-

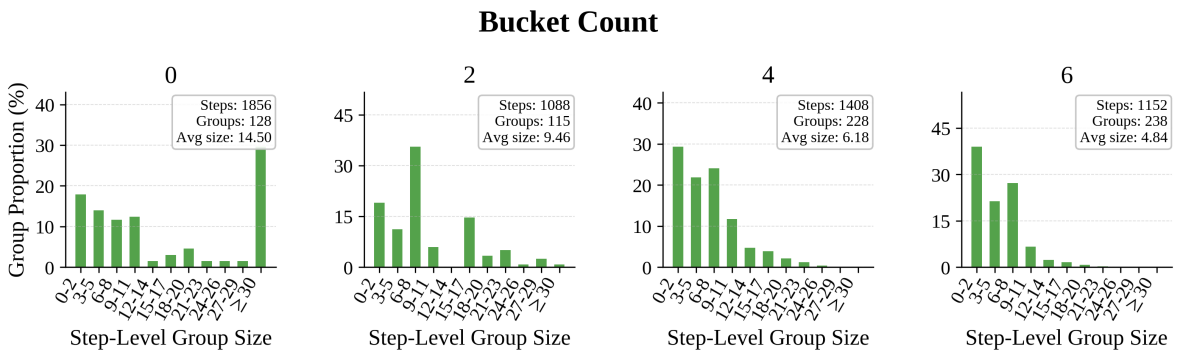

Figure 4: State cluster size distribution under different progress bucket counts.

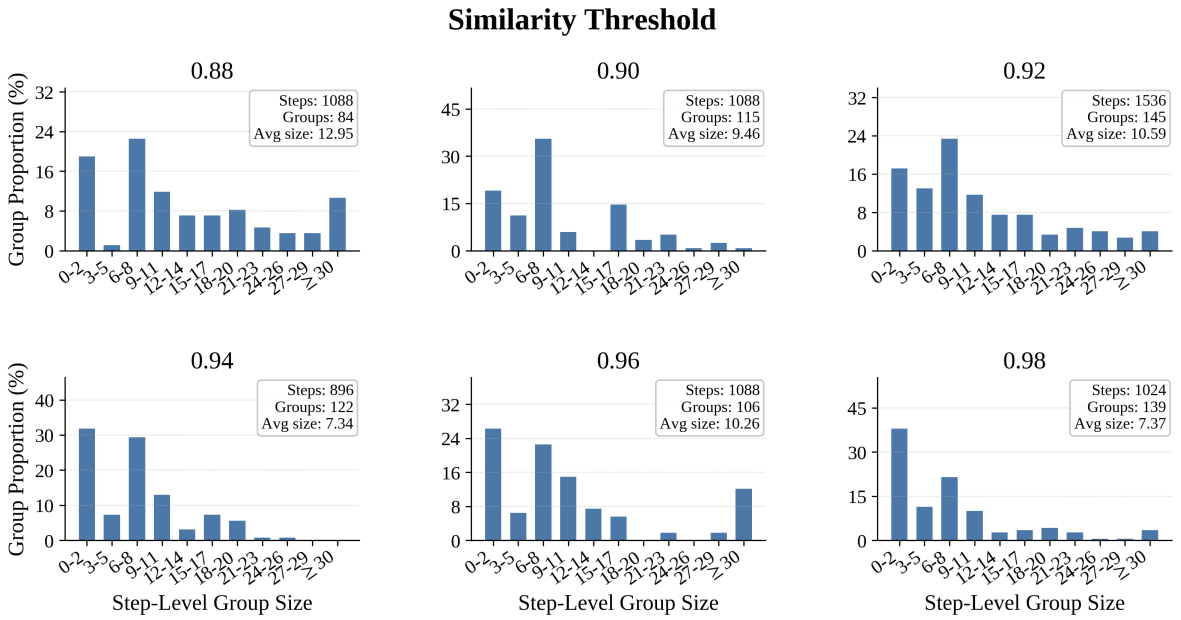

Figure 5: State cluster size distribution under different similarity thresholds.

tering on WebShop validation records. The plotting script bins the raw cluster sizes into `0-2`, `3-5`, . . ., and ≥ 30, and divides the cluster count in each bin by the `Total step-level group count`. Therefore, these figures reflect the distribution over the number of state clusters rather than the distribution over the number of steps. In general, a high ratio of small clusters indicates overly fine grouping and insufficient intra-cluster comparison samples. Conversely, a high ratio of large clusters, especially clusters in the $\geq$ 30 bin, suggests overly coarse grouping or frequent hits to the cluster capacity limit, which can mix states from different decision stages.

Similarity Threshold. The default threshold $\tau = 0.90$ yields the most balanced distribution. It has 115 state clusters with an average size of 9.46. Of these, 66.09% fall in the `0-8` bin. Large clusters (`>=21`) make up only 9.57%, and those `>=30` account for just 0.87%. A lower threshold ($\tau = 0.88$) easily merges states. The cluster count drops to 84, and the average size rises to 12.95. The `>=30` large clusters reach 10.71%, showing clear over-merging. Higher thresholds generally push the distribution toward smaller clusters. For example, at $\tau = 0.94$ and $\tau = 0.98$, the ratio of small to medium-small clusters (`0-11`) reaches 81.97% and 81.29%, respectively. Their average cluster size is about 7.3. This result shows that a higher similarity threshold is not always better. $\tau = 0.90$ strikes a better balance between avoiding large cluster mixing and retaining enough intra-cluster samples.

History Window Size. When $K = 0$, local variation information is missing, making the grouping clearly too coarse. There are only 85 state clusters, and the average size reaches 17.32. Large clusters (`>=21`) account for 45.88%, with those `>=30`

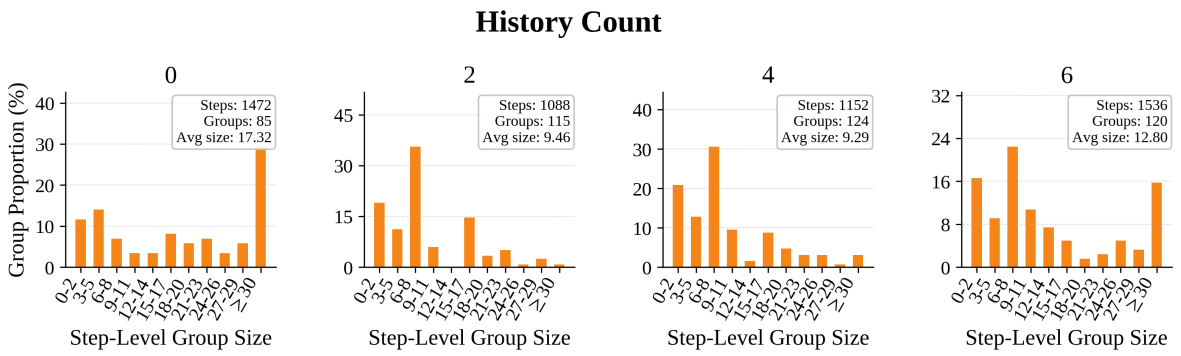

Figure 6: State cluster size distribution under different history window sizes.

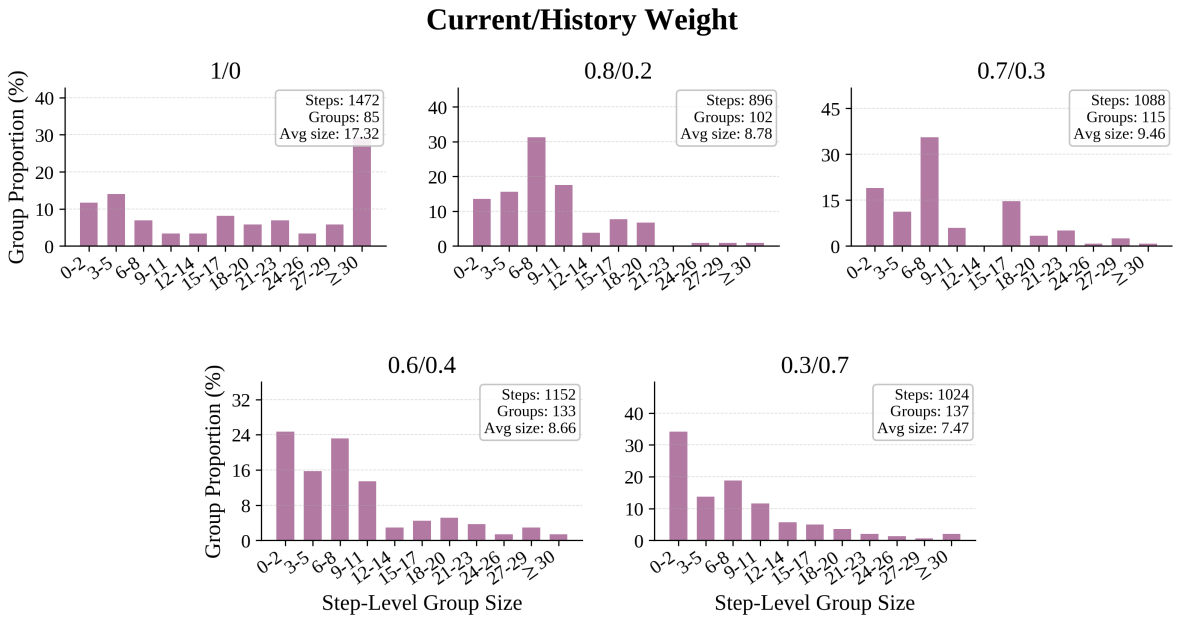

Figure 7: State cluster size distribution under different Current/Innovation branch weights.

making up 29.41%. After adding a short history ($K = 2$), the average size drops to 9.46, and the `>=30` ratio falls to 0.87%. This shows that the local variation branch effectively separates states that look similar but have different dynamic contexts. Further increasing the window size is unstable. The distribution for $K = 4$ is similar to $K = 2$, but its `>=30` ratio rebounds to 3.23%. For $K = 6$, the average size rises to 12.80, and the ratio of `>=21` clusters reaches 26.67%. This indicates that a short history already provides effective dynamic information. A history that is too long might introduce noise that is weakly related to the current decision, thereby weakening state differentiation.

Progress Bucket Count. When we disable progress bucketing ($B = 0$), cross-stage steps are more easily merged. The average cluster size is 14.50. Large clusters (`>=21`) account for 34.38%, and those `>=30` account for 29.69%. Enabling the default $B = 2$ drops the average size to 9.46 and the `>=30` ratio to 0.87%. This shows that the relative progress constraint noticeably reduces cross-stage mixing. Increasing the bucket count further fragments the state clusters. For $B = 4$ and $B = 6$, the number of state clusters rises to 228 and 238, respectively. Their average cluster sizes drop to 6.18 and 4.84. Also, the ratio of `0-8` small clusters reaches 75.44% and 87.82%, respectively. Thus, the main role of bucketing is to prevent over-merging. However, too many buckets lower the sample utilization inside each bucket, making local advantage estimation rely more on small cluster statistics.

Current/Innovation Branch Weight. Using only the current state branch (1.0/0.0) yields a distribution identical to having no historical information. The average size is 17.32, and `>=30` large clusters make up 29.41%. This shows that the

original observation semantics alone cannot stably distinguish local states in long-horizon interactions. Adding the local variation branch drops the large cluster ratio significantly. The average cluster sizes for 0.8/0.2, 0.7/0.3, and 0.6/0.4 are 8.78, 9.46, and 8.66, respectively. The `>=30` ratio for all three is under 1.50%. For the default 0.7/0.3, the `0-8` small cluster ratio is 66.09%. It still retains some `9-20` medium clusters (24.35%), matching a more stable task success rate. By contrast, the 0.3/0.7 setting overemphasizes local variation. Its average size drops further to 7.47, and the small cluster ratio rises. This might make the grouping too sensitive to short-term changes and lower semantic stability.

Overall, the four distributions jointly support the conclusion in the main text. SAGE's key is not just generating more small clusters. It is maintaining a moderate granularity between state conditionality and intra-cluster statistical stability. The default setup ($\tau = 0.90, K = 2, B = 2, \omega_{\text{cur}}/\omega_{\text{inn}} = 0.7/0.3$) usually avoids the state mixing caused by `>=30` large clusters. It also avoids creating overly many small clusters like an excessively high threshold or too many buckets. Thus, it is better suited for building stable step-level local relative advantages.