COMPGRAPHRL: CREDIT ASSIGNMENT OVER COMPRESSED SEMANTIC STATE GRAPHS FOR MOBILE GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Online reinforcement learning (RL) in open-ended environments has improved the planning and navigation capabilities of mobile graphical user interface (GUI) agents. However, long-horizon GUI tasks typically yield sparse terminal rewards, creating a severe temporal credit-assignment problem. Existing approaches either assign credit independently within trajectories or detect repeated states through exact observation or environment-state matching. As a result, they fail to identify semantically equivalent states under variations in rendering, dynamic content, and interaction history. We introduce CompGraphRL, a credit-assignment framework based on compressed semantic state graphs. CompGraphRL uses a vision-language model (VLM) to encode each observation jointly with the preceding action and interaction history, merging functionally equivalent states across trajectories into shared graph nodes. We further generalize Generalized Advantage Estimation (GAE) to graph-structured interactions by separately estimating node-level and action-conditioned edge advantages. This design propagates delayed rewards through the graph while providing fine-grained, low-variance signals for policy optimization. Large-scale online RL experiments on MobileGym, with cross-domain evaluation on AndroidWorld and MobileWorld, show that CompGraphRL consistently outperforms sequence-level and graph-based credit-assignment baselines across Qwen3.5 and GUI-OWL-1.5 backbones. With Qwen3.5-27B, it improves MobileGym success from 20.7% to 37.1% and the mean success rate on the two out-of-domain benchmarks from 31.9% to 44.9%. Further analysis shows that a node-merging rate of approximately 15% effectively balances cross-trajectory information sharing and state discriminability.

1 INTRODUCTION

Recent vision–language models (VLMs) have shown strong zero-shot performance in web and mobile graphical user interface (GUI) interaction (Qwen Team, 2026a; Yang et al., 2025; Zeng et al., 2026). However, most prior work adapts pre-trained VLMs for GUI control using static expert trajectories through supervised fine-tuning or action-level reinforcement learning (RL) (Ye et al., 2025; Xu et al., 2026; Yan et al., 2025; Cao et al., 2026). Such offline approaches are limited to fixed datasets: they cannot learn from execution outcomes or adapt to unseen states. This limitation is particularly acute in open-ended GUI environments, which require long-horizon reasoning over evolving states and unfamiliar interaction flows, where early errors can lead to irreversible failures. Online RL addresses these challenges by enabling agents to improve through closed-loop exploration, optimizing policies from complete trajectories and task-level feedback under the current policy. Recent work combining asynchronous training with policy-optimization algorithms such as GRPO and PPO has substantially improved end-to-end GUI task success (Xu et al., 2025; Zhang et al., 2025; Zhou et al., 2026; Qin et al., 2025; Wang et al., 2025; Zhou et al., 2025).

Nevertheless, online RL remains notoriously challenging for long-horizon mobile GUI tasks, where agents must navigate vast, structured action spaces over extended trajectories guided solely by sparse terminal feedback. Across standard open-ended benchmarks such as AndroidWorld (Rawles et al., 2025), MobileWorld (Kong et al., 2026), and MobileGym (Wu et al., 2026), an agent often executes dozens of interaction steps before receiving any task-level reward, precipitating frequent invalid ac-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

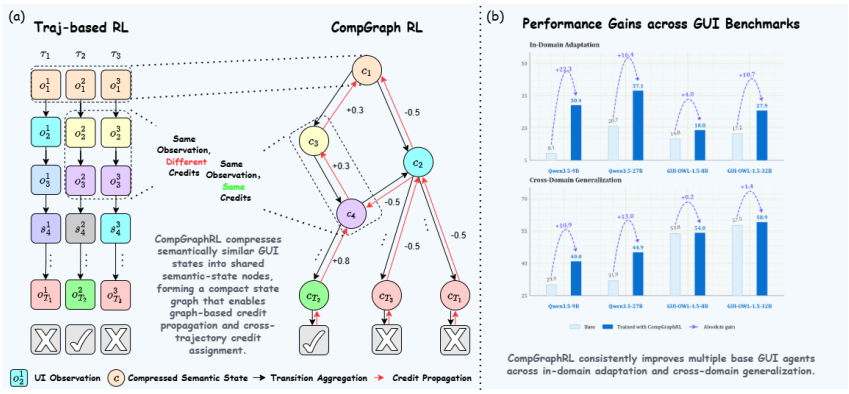


Figure 1: Overview of CompGraphRL. (a) Standard trajectory-based credit assignment is restricted to individual trajectories. CompGraphRL maps semantically similar GUI states from different trajectories to shared nodes in a compressed semantic state graph, enabling credit propagation across trajectories. (b) This graph-based credit assignment improves mobile GUI agent performance under both in-domain adaptation and cross-domain generalization.

tions and severe temporal credit assignment bottlenecks. Furthermore, identical early exploration steps can receive conflated trajectory-level credit due to downstream compounding errors. While recent fine-grained and graph-based RL methods alleviate this issue by aggregating recurrent states and propagating credit over transition graphs (Li et al., 2026; Feng et al., 2026; Xie et al., 2026; Wang et al., 2026), they predominantly rely on exact observations, explicit environment states, or identical action sequences to identify state equivalence. Such assumptions break down in mobile environments: semantically equivalent states often manifest as visually distinct screens due to variations in rendering, resolution, themes, or dynamic content, whereas visually congruent screens can correspond to fundamentally disparate latent states. Consequently, constructing robust semantic state abstractions for faithful credit assignment remains an open, pivotal challenge in mobile GUI RL.

To address the issue of trajectory-isolated credit assignment in online RL for mobile GUIs, we propose CompGraphRL, a framework built upon compressed semantic state graphs. As illustrated in Figure 1(a), CompGraphRL encodes each interaction step—comprising the current observation, previous action, and history—into a compact representation using a vision-language model (VLM). Crucially, semantically equivalent GUI states across disjoint trajectories are merged into shared nodes, constructing a compressed transition graph that aggregates environment dynamics and reward signals while preserving action-specific outcomes. CompGraphRL then conducts graph-based credit propagation, estimating state values and propagating delayed feedback backward along transitions to generate edge-level learning signals. Consequently, successful trajectories reinforce shared predecessor states and effective actions, while alternative actions receive outcome-dependent adjustments. By bridging equivalent states across trajectories without conflating distinct action effects, CompGraphRL effectively alleviates credit sparsity and variance.

We train our agents within the open-ended environment MobileGym (Wu et al., 2026) and evaluate them end-to-end on AndroidWorld (Rawles et al., 2025) and MobileWorld (Kong et al., 2026). Across various backbone architectures, CompGraphRL consistently enhances both in-domain adaptation and cross-domain generalization (Figure 1(b)). Specifically, for Qwen3.5-27B, task success rates increase from 20.7% to 37.1% on MobileGym and from 31.9% to 44.9% across out-of-domain benchmarks. These empirical gains demonstrate that compressed semantic graphs effectively propagate learning signals across functionally related GUI trajectories.

Our main contributions are as follows:

- We propose CompGraphRL, which constructs semantic state graphs from noisy GUI interaction trajectories using a VLM-based state compressor, and then performs graph-structured credit assignment over the compressed graphs.

- We extend Generalized Advantage Estimation (GAE) to compressed state graphs and explicitly separate node and edge advantages, enabling finer-grained credit assignment for GUI operations beyond single linear trajectories.
- We conduct large-scale online RL experiments in MobileGym and evaluate transfer to AndroidWorld and MobileWorld. Across Qwen3.5 and GUI-OWL backbones, CompGraphRL improves in-domain performance and generalizes effectively across domains. Further analysis reveals that compression quality is crucial: excessive merging or splitting degrades RL performance, whereas a node-merging rate of approximately 15% balances state discriminability with cross-trajectory information sharing.

2 Related Work

GUI Agents GUI agents have evolved from single-step instruction grounding to multi-turn planning and long-horizon interaction, enabled by advances in data, model adaptation, and scalable training environments (Wang et al., 2024; Qin et al., 2025; Gu et al., 2025; Liu et al., 2024). Early approaches adapted pretrained vision–language models using static expert trajectories, such as AndroidControl and GUI Odyssey, via supervised fine-tuning, action-level RL, or continual pretraining (Li et al., 2024; Lu et al., 2025; Ye et al., 2025; Xu et al., 2026; Yan et al., 2025; Cao et al., 2026; Zhou et al., 2025; Venus Team et al., 2026). Recent interactive benchmarks and simulators—including AndroidWorld, MobileWorld, AndroidDaily, and MobileGym—have shifted the focus to dynamic, open-ended settings where agents learn from execution outcomes (Rawles et al., 2025; Kong et al., 2026; Sui et al., 2026; Wu et al., 2026). Accordingly, online RL methods, including UI-Tars-2, mobileRL, AgentRL, and Qwen-UI-Agent, improve multi-turn decision-making through closed-loop rollouts, asynchronous sampling, and data flywheels (Wang et al., 2025; Xu et al., 2025; Zhang et al., 2025; Zhou et al., 2026). However, long-horizon tasks yield sparse terminal rewards, and existing methods typically optimize trajectory-level signals without assigning shared credit to semantically equivalent states across rollouts. CompGraphRL addresses this limitation through online learning in an open-ended mobile environment. To our knowledge, it is the first GUI-agent training framework for open-ended environments to introduce graph-based credit assignment: a compressed semantic state graph links functionally equivalent GUI states across trajectories and propagates delayed outcomes to action-specific transitions, yielding denser, lower-variance learning signals without additional step-level reward annotation.

Credit Assignment for LLM Agents RL is central to enabling LLM agents to adapt in open-ended environments (Zhou et al., 2023; Webb et al., 2025; Liu et al., 2026c). Group-based methods, including GRPO, RLOO, DAPO, and GSPO (Shao et al., 2024; Hu et al., 2025; Yu et al., 2026; Zheng et al., 2025), improve complex reasoning but assign credit at the trajectory level, resulting in sparse, high-variance supervision for long-horizon, multi-turn tasks. Fine-grained approaches provide denser feedback through implicit process rewards, or learned step-level critics (Cui et al., 2025; Li et al., 2026; Liu et al., 2026d), but often require complex reward or critic model training. Graph-based methods such as GiGPO (Feng et al., 2026) and G2PO (Wang et al., 2026) evaluate local actions or reconstruct state-transition graphs; however, they identify shared states mainly through repeated observations or environment-provided identities, limiting credit sharing across visually distinct but functionally equivalent GUI states. CompGraphRL addresses these limitations in two ways: a VLM-based compressor maps semantically equivalent visual states to shared graph nodes without repeated observations, and a lightweight critic combined with graph-structured GAE propagates outcomes across these nodes to derive transition-level advantages. This yields denser supervision and lower-variance credit estimates without relying on costly critic models.

3 CompGraphRL

3.1 Problem Formulation

We model mobile GUI interaction as a finite-horizon partially observable Markov decision process (POMDP), $M = (\mathcal{S}, \mathcal{A}, P, R, \Omega, O, H)$. The latent state $\mathcal{S}$ comprises observable UI elements and hidden application states. The discrete action space $\mathcal{A}$ includes atomic GUI operations: `type`, `swipe`, `click`, `drag`, `long press`, `home`, `back`, and `terminate`. The transition model $P(s_{t+1} \mid s_t, a_t)$ captures the stochastic dynamics of the Android system and applications. The

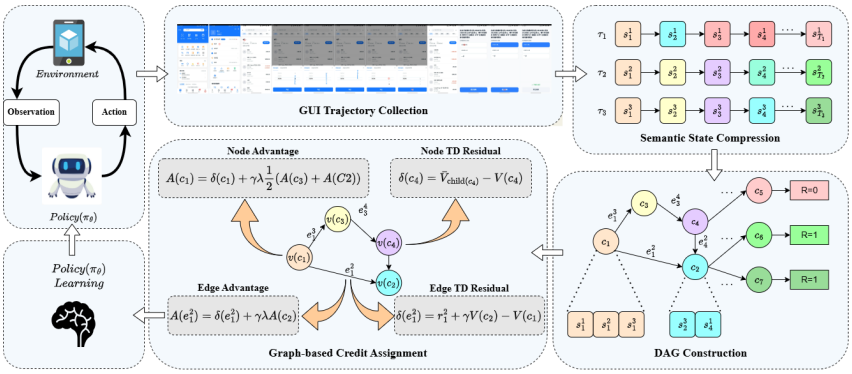


Figure 2: Overview of CompGraphRL. CompGraphRL first encodes each interaction state into a compact VLM-based semantic representation. It then constructs a compressed directed state graph by merging functionally equivalent GUI states across trajectories, while preserving action-induced transitions as graph edges. On the resulting graph, CompGraphRL estimates node values to share return information among equivalent states and derives edge-level advantages to assign credit to specific transitions. These graph-structured signals propagate sparse terminal feedback across trajectories and provide fine-grained turn-level supervision for policy optimization.

observation space Ω contains visual inputs and interaction history, with observation model $O(o_t \mid s_t)$. At step t, the agent receives o_t and selects a_t according to $\pi(a_t \mid o_t)$. Given o_0, an episode generates $\tau = (o_0, a_0, \ldots, o_T, a_T)$, where $T \leq H$. It terminates upon executing `terminate` in a successful state or reaching the horizon. We use a sparse terminal reward, $R(\tau) \in \{0, 1\}$, indicating failure or success, respectively.

In the following section, we present CompGraphRL, a fine-grained credit-assignment framework for multi-turn online RL in GUI agents. As shown in Figure 2, it consists of three components. First, a semantic GUI-state encoder maps each interaction step to a compact embedding that captures visual feedback, prior actions, and interaction history. Second, a controlled graph-construction module merges functionally equivalent states across trajectories under explicit merge budgets and temporal constraints, producing a compressed directed state graph without requiring exact observation matches. Finally, credit propagation over this graph estimates node values and edge-level advantages for policy optimization.

3.2 Semantic State Compression

GUI tasks require an agent to execute software operations, enter information, and interpret visual feedback. At each step, the agent outputs an executable action, which the environment executes (e.g., click or swipe) and then returns an updated page state after a fixed delay. We define the state at step t as $S_t = (a_{t-1}, o_t, h_{t-1})$, where a_{t-1} is the previous action, o_t is the current visual observation, and h_{t-1} is the interaction history up to step $t-1$. The initial state is $S_0 = (q, o_0)$, where q is the user query and o_0 is the initial observation. Unlike conventional observation-only representations, our state formulation incorporates action and history. Existing graph-based RL methods commonly merge states using identical observations or environment-provided identifiers. In contrast, CompGraphRL distinguishes semantically different decision stages with similar visual appearances, such as revisiting a page after submitting a form versus viewing it before entering task-relevant information.

State compression is a central component of CompGraphRL. Exact observation matching is unreliable in GUIs because functionally equivalent states may differ in layout, widget positions, resolution, themes, or dynamic content. Conversely, visually similar screens may represent different task stages depending on prior actions and history, while text-only summaries often omit spatial and functional cues necessary for action selection. CompGraphRL therefore employs a VLM-based encoder to jointly represent visual appearance, interface semantics, previous actions, and interaction history, enabling function-level state equivalence under GUI-specific perturbations while reducing state-space sparsity. Formally, the parametric encoder produces $z_t = f_\phi(S_t)$, where $f_\phi(\cdot)$ is param-

eterized by ϕ and $z_t \in \mathcal{Z}$ is the compressed state embedding. We measure the similarity between embeddings using cosine similarity $s_{ij} = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\|\mathbf{z}_i\|_2 \|\mathbf{z}_j\|_2}$.

3.3 Controlled State Graph Construction

Given semantic GUI state representations, CompGraphRL constructs the state graph through controlled cross-trajectory merging. The goal of this module is not merely to cluster similar observations, but to decide which interaction states should share a functional node while preserving distinct decision stages. Given a query, we assume that the model samples multiple complete execution trajectories $\tau_k = \{(o_0^k, a_0^k, o_1^k, a_1^k, \ldots, o_{T_k}^k)\}$, where k indexes the trajectory. Unlike approaches that process trajectories independently or group only exact repeated observations, our method merges semantically equivalent GUI states across trajectories using compressed state representations, yielding a unified semantic transition structure.

Specifically, we cluster state representations and define the cluster associated with state s_i as $c_i = \{s_t \mid s_t \sim s_i\}$, where $\sim$ denotes state similarity. Each cluster c_i is treated as a node in the global structure. If the agent executes action a from a state in cluster c_i and transitions to a state in cluster c_j, we introduce a directed edge $e_i^j : c_i \xrightarrow{a} c_j$. Accordingly, the successor set of c_i is defined as $\mathrm{Succ}(c_i) = \{c_j \mid \exists a,\ c_i \xrightarrow{a} c_j\}$, and the predecessor set of a non-initial node c_j is $\mathrm{Pred}(c_j) = \{c_i \mid \exists a,\ c_i \xrightarrow{a} c_j\}$. By aggregating transition relations across all sampled trajectories, we obtain a semantic state-transition graph $\mathcal{T} = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C}$ denotes the set of clustered GUI state nodes and $\mathcal{E}$ denotes the set of action-induced directed transition edges.

To adaptively control state-compression granularity, we specify a target merge rate ρ_{target} rather than a fixed similarity threshold. For a state-transition graph with N states, including N_{init} trajectory-initial states, the merge budget is $B_{\mathrm{merge}} = \mathrm{round}((N - N_{\mathrm{init}})\rho_{\mathrm{target}})$. We rank candidate merges using an adjusted similarity score that combines VLM embedding similarity with temporal constraints. To enable backward credit propagation on $\mathcal{T}$, we preserve acyclicity throughout graph construction: a merge is accepted only if collapsing its clusters introduces no directed cycle; otherwise, it is rejected. Consequently, $\mathcal{T}$ remains a directed acyclic graph, allowing node and edge advantages to be computed recursively from terminal nodes without cyclic dependencies. Therefore, ρ_{target} defines an upper bound, rather than an exact compression rate. As shown in Section 4.4, target merge rates of 10–20% achieve the best training performance, whereas excessive merging or splitting is detrimental. Details of the dynamic merge-rate controller and adjusted scoring rule are provided in Appendix A.2.

3.4 Graph-based Credit Assignment

We generalize Generalized Advantage Estimation (GAE) to graph structures to achieve fine-grained credit assignment across state transitions, leveraging the compressed semantic embeddings $\mathcal{Z}$ and the transition graph $\mathcal{T} = (\mathcal{C}, \mathcal{E})$. Specifically, a lightweight value model is trained on Z to predict individual state values, with each cluster node $c_i \in \mathcal{C}$ assigned the mean value of its constituent states. We then propagate credit backward over $\mathcal{T}$, computing Temporal Difference (TD) residuals and advantages at the node level before refining them to the edge level. This two-stage formulation distinguishes the outcomes of distinct actions within the same semantic state, yielding lower-variance, more reliable advantage estimates. The algorithmic details are formulated below.

We first compute the temporal-difference (TD) residual for each node. For a non-terminal node c_i, the average bootstrapped value over its successor nodes is defined as

$$\bar{V}_{\mathrm{succ}(c_i)} = \frac{1}{|\mathrm{Succ}(c_i)|} \sum_{c_j \in \mathrm{Succ}(c_i)} \left(r_i^j + \gamma V(c_j) \right),$$

where r_i^j denotes the average immediate reward obtained when transitioning from node c_i to its successor node c_j, and γ is the discount factor. The TD residual at node c_i is then defined as

$$\delta(c_i) = \bar{V}_{\mathrm{succ}(c_i)} - V(c_i).$$

Given the node-wise TD residuals, we generalize trajectory-level GAE to graphs. Analogous to computing $\bar{V}_{\mathrm{succ}(c_i)}$ by averaging over successor nodes, we recursively compute graph-structured

advantages from the mean successor advantage:

$$A(c_i) = \delta(c_i) + \gamma\lambda \frac{1}{|\mathrm{Succ}(c_i)|} \sum_{c_j \in \mathrm{Succ}(c_i)} A(c_j),$$

where λ is the GAE decay parameter. For a terminal node c_v, no successor nodes are available for bootstrapping. Therefore, we directly compute its advantage using the final return $R(v)$ of the corresponding trajectory $A(c_v) = R(v) - V(c_v)$.

In trajectory-level GAE, a node at step t and its corresponding edge receive the same advantage. In an aggregated graph, however, multiple outgoing edges may share a source node but yield different future outcomes. We therefore distinguish node advantage, the expected advantage averaged over subsequent actions, from edge advantage, the advantage of selecting a specific action. For an edge e_i^j from node c_i to successor node c_j, we define the temporal-difference residual as follows:

$$\delta(e_i^j) = r_i^j + \gamma V(c_j) - V(c_i),$$

where r_i^j denotes the average immediate reward over all transition samples aggregated into edge e_i^j and edge advantage is then defined recursively as follows:

$$A(e_i^j) = \delta(e_i^j) + \gamma\lambda A(c_j).$$

Finally, we assign each edge advantage $A(e_i^j)$ to all original transitions aggregated within that edge, yielding the step-level advantage A_i^j for the j-th step of trajectory i. Following GRPO, we optimize

$$\mathcal{L}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{j=1}^{T_i}\min\left(\rho_{i,j}(\theta)A_i^j,\ \mathrm{clip}\left(\rho_{i,j}(\theta), 1-\epsilon, 1+\epsilon\right)A_i^j\right) + \beta D_{\mathrm{KL}}\left(\pi_\theta \,\|\, \pi_{\theta_{\mathrm{ref}}}\right),$$

where$\rho_{i,j}(\theta) = \frac{\pi_\theta\left(a_i^j|s_i^j\right)}{\pi_{\theta_{\mathrm{sample}}}\left(a_i^j|s_i^j\right)}$is the action-level importance ratio at step j of trajectory i. The KL regularizer constrains the updated policy toward the reference policy. The value model used to estimate edge advantages is pretrained on a subset of collected trajectories and frozen during policy optimization; its objective is provided in Appendix A.4.

4 Experiments

4.1 Experimental Setting

We train on 160 MobileGym tasks and evaluate on the MobileGym test set (256 tasks), AndroidWorld (Rawles et al., 2025), and MobileWorld (Kong et al., 2026). Our agents use Qwen3.5-9B/27B (Qwen Team, 2026a) and GUI-OWL-1.5-8B/32B (Xu et al., 2026) as backbones. We compare CompGraphRL with GSPO (Zheng et al., 2025), a stable sequence-level optimizer for long-horizon, multi-turn GUI tasks, and two fine-grained graph-based credit-assignment baselines: GiGPO (Feng et al., 2026), which groups decisions by shared local states to estimate action preferences, and G2PO (Wang et al., 2026), which aggregates values over reconstructed state-transition graphs. Because exact state repetitions are virtually absent in GUI trajectories due to rendering variation and dynamic visual content, exact observation matching would leave GiGPO and G2PO with almost no shared states. We therefore follow an observation-based approximation for both baselines: states are treated as identical and merged when the similarity between their observation images exceeds the matching threshold. We evaluate GiGPO and G2PO on the Qwen3.5-9B and Qwen3.5-27B backbones using the same MobileGym rollouts, reward shaping, prompts, and optimization settings as CompGraphRL; no CompGraphRL semantic compressor or learned value model is supplied to either baseline. We exclude critic-based PPO because it requires critic pretraining (Yuan et al., 2025); when trained from scratch in MobileGym, it degraded and ultimately collapsed. For CompGraphRL, we freeze Qwen3-VL-4B-Instruct (Yang et al., 2025) as a state compressor and represent each state using its final-token embedding. A lightweight two-layer LSTM value model processes these embeddings and is frozen after brief initial training. We apply GUI-specific reward shaping: terminal rewards are $+4$ for success and -2 for failure; invalid or repeated actions receive -0.1; malformed outputs receive -4.

Model	MobileGym	AndroidWorld	MobileWorld	Avg
(*Open-weight GUI agents*)				
Qwen3-VL-8B-Instruct (Yang et al., 2025)	9.3^{*}	47.6	6.8	21.2
Qwen3-VL-32B-Instruct (Yang et al., 2025)	16.3^{*}	57.3	10.5	28.0
UI-Tars-1.5-8B (ByteDance Seed, 2025)	13.8	64.2	28.0	35.3
Step-GUI-8B (Yan et al., 2025)	13.8^{*}	67.7	–	
Mai-UI-8B (Zhou et al., 2025)	16.8^{*}	70.7	27.5	38.3
UI-Venus-1.5-8B (Venus Team et al., 2026)	15.4	**73.7**	12.4	33.8
ForgeOwl-8B (Liu et al., 2026b)	–	69	29.8	
Qwen3.5-9B (Qwen Team, 2026a)				
Base	$8.1^{*}_{\pm 1.5}$	$52.3^{*}_{\pm 0.9}$	$7.5^{*}_{\pm 0.7}$	22.6
GSPO	$23.8_{\pm 1.2}$	$58.6_{\pm 1.2}$	$12.6_{\pm 2.0}$	31.6
GiGPO	$22.4_{\pm 1.5}$	$56.6_{\pm 0.5}$	$13.0_{\pm 0.5}$	30.6
G2PO	$23.9_{\pm 2.4}$	$53.3_{\pm 0.5}$	$16.1_{\pm 0.8}$	31.1
CompGraphRL	$30.4_{\pm 0.2}$	$63.3_{\pm 0.5}$	$18.2_{\pm 0.7}$	37.3
Qwen3.5-27B (Qwen Team, 2026a)				
Base	$20.7^{*}_{\pm 1.5}$	$53.0^{*}_{\pm 0.8}$	$10.7^{*}_{\pm 0.8}$	28.1
GSPO	$34.0_{\pm 1.1}$	$62.0_{\pm 2.1}$	$15.4_{\pm 0.8}$	37.1
GiGPO	$34.5_{\pm 0.7}$	$64.6_{\pm 0.5}$	$14.8_{\pm 0.5}$	37.9
G2PO	$34.9_{\pm 0.6}$	$60.0_{\pm 1.2}$	$14.2_{\pm 0.5}$	36.4
CompGraphRL	$\mathbf{37.1_{\pm 2.8}}$	$71.0_{\pm 0.5}$	$18.8_{\pm 0.7}$	42.3
GUI-OWL-1.5-8B-Instruct (Xu et al., 2026)				
Base	$14.8^{*}_{\pm 0.8}$	$68.0^{*}_{\pm 0.8}$	$39.5^{*}_{\pm 1.0}$	40.7
GSPO	$17.2_{\pm 0.5}$	$67.3_{\pm 0.5}$	$37.5_{\pm 0.6}$	40.6
CompGraphRL	$18.8_{\pm 0.9}$	$69.3_{\pm 1.8}$	$38.6_{\pm 0.8}$	42.2
GUI-OWL-1.5-32B-Instruct (Xu et al., 2026)				
Base	$17.2^{*}_{\pm 0.5}$	$68.9^{*}_{\pm 0.7}$	$46.1^{*}_{\pm 1.1}$	44.1
GSPO	$26.8_{\pm 0.8}$	$68.0_{\pm 0.8}$	$45.2_{\pm 0.8}$	46.6
CompGraphRL	$27.9_{\pm 0.7}$	$70.0_{\pm 0.8}$	$\mathbf{47.8_{\pm 1.3}}$	**48.6**

Table 1: Performance comparison on the MobileGym, AndroidWorld, and MobileWorld benchmarks. We report success rates (%) for each benchmark, averaged over three random seeds. Results marked with "$*$" were evaluated by us. The best results are highlighted in bold.

We use VERL's asynchronous framework (Sheng et al., 2024), with vLLM (Kwon et al., 2023) for rollout generation and FSDP (Paszke et al., 2019) for training. All reinforcement learning methods use identical hyperparameters and rollout prompts. We sample $N = 8$ trajectories per query and use a learning rate of 5×10^{-7}. We select the checkpoint with the best validation performance for final evaluation. Each task is evaluated independently across three runs, and we report the mean. To control for seed-dependent variation in MobileGym difficulty, we use the same three evaluation seeds across all experiments. Implementation details and hyperparameters are provided in Appendix A.4. Unless otherwise stated, the compression strategy analysis, merge ratio analysis, and ablation study are all conducted with Qwen3.5-9B as the backbone policy on MobileGym.

4.2 Main Results

Table 1 shows that CompGraphRL achieves the strongest overall performance among open-source methods, yielding the best average score for every backbone and outperforming the corresponding base models by 14.7, 14.2, 1.5, and 4.5 percentage points and GSPO by 5.7, 5.2, 1.6, and 2.0 points, respectively. These gains span both in-domain adaptation and cross-domain generalization: CompGraphRL improves MobileGym by up to 22.3 points and, for the Qwen3.5 backbones, improves AndroidWorld and MobileWorld by up to 18.0 and 10.7 points, respectively. Its larger gains on Qwen3.5 than on GUI-OWL-1.5 suggest particular effectiveness in eliciting transferable GUI capabilities from general-purpose backbones.

Finally, comparisons with graph-based credit-assignment methods isolate the benefit of semantic state compression. CompGraphRL consistently outperforms GiGPO and G2PO across model scales, improving average performance over the stronger of the two baselines by 6.2 and 4.4 percentage points with Qwen3.5-9B and Qwen3.5-27B, respectively. In contrast, GiGPO and G2PO provide limited and inconsistent gains over GSPO, likely because image-similarity-based grouping can sep-

Method	Partition-level					Pairwise-level			MobileGym
	$ARI \uparrow$	$NMI \uparrow$	$VI_{norm} \downarrow$	$Hom. \uparrow$	$Comp. \uparrow$	$Precision \uparrow$	$Recall \uparrow$	$F1 \uparrow$	$Succ. \uparrow$
LTC	0.416	0.763	0.334	**1.000**	0.621	**1.000**	0.307	0.446	26.3
ISC	0.556	0.784	0.229	0.807	0.795	0.650	0.706	0.616	17.6
IAC	0.584	0.804	0.221	0.864	0.773	0.725	0.645	0.635	19.2
TSC	0.244	0.473	0.442	0.375	0.682	0.276	**0.728**	0.380	13.8
RTC	0.500	0.783	0.293	0.975	0.661	0.934	0.403	0.532	27.6
VSC	**0.676**	**0.845**	**0.190**	0.966	**0.757**	0.919	0.594	**0.705**	**29.6**

Table 2: Performance comparison across compression strategies on MobileGym with Qwen3.5-9B as the backbone. Higher values indicate better performance for ARI, NMI, homogeneity, completeness, pairwise precision, pairwise recall, pairwise F1 score, and MobileGym success rate, whereas lower normalized variation of information (VI) values are preferable.

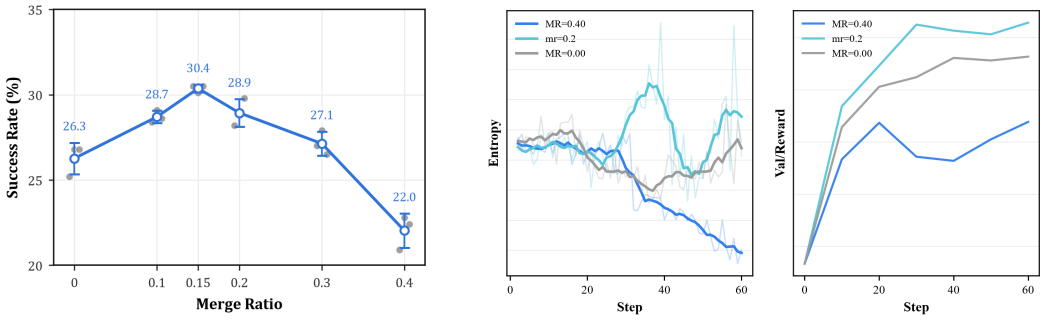
arate functionally equivalent states or merge visually similar states from distinct task stages. By jointly modeling visual semantics, preceding actions, and interaction history, CompGraphRL enables more reliable state compression and cross-trajectory credit propagation.

4.3 Compression Strategy Analysis

We evaluate six compression strategies using human-annotated state partitions from 240 trajectories sampled across 30 MobileGym tasks. The baselines cover trajectory-structure-, visual-, action-, and text-based aggregation, while our VLM State Compression (VSC) uses vision-language representations. We assess agreement with human annotations using partition-level and pairwise metrics and additionally report downstream MobileGym success. All similarity-based methods use the same threshold of 0.95; Appendix A.5 provides the annotation protocol, complete method definitions, implementation details, and formal metric definitions.

As shown in Table 2, VSC achieves the best ARI, NMI, VI_{norm}, Pairwise-F1, and MobileGym success, demonstrating the closest agreement with human merge decisions and the strongest downstream performance. Error analysis shows that LTC and RTC tend to over-split states, whereas ISC, IAC, and TSC tend to over-merge distinct states. Although both errors are harmful, over-merging causes substantially greater downstream degradation, highlighting the importance of conservative semantic merging that preserves distinct decision states.

4.4 Merge Ratio Analysis



(a) (b)

Figure 3: Effect of merge ratio on downstream performance and training dynamics using Qwen3.5-9B. (a) MobileGym Success Rate (%) across target merge ratios, averaged over three seeds. Moderate compression performs best: ratios of 0.10–0.20 achieve the strongest results, peaking at $\rho_{\text{target}} = 0.15$. (b) Training curves for representative ratios. Insufficient merging yields stable but limited gains, whereas aggressive merging rapidly collapses entropy, suggesting that over-merging confounds credit assignment by combining semantically distinct GUI states.

Our prior analysis shows that compression strongly influences downstream performance. We therefore examine the appropriate degree of semantic merging using the same VSC-based state com-

pressor. As shown in Figure 3a, performance varies non-monotonically with the target merge ratio. Moderate merging consistently outperforms both no merging and aggressive merging: ratios of 0.1–0.2 perform best, and $\rho_{\text{target}} = 0.15$ achieves the highest Success Rate (%) (30.4%). Without merging, equivalent states remain isolated across trajectories, limiting information and reward sharing. In contrast, increasing the ratio to $\rho_{\text{target}} = 0.4$ lowers Success Rate (%) to 22.0%.

Figure 3b explains this pattern. Insufficient merging yields stable but weaker training because functionally equivalent states remain fragmented, resulting in trajectory-local advantage estimates and limited reward reuse. Over-merging is more harmful: high merge ratios cause rapid entropy collapse, suggesting that states with distinct optimal actions or future outcomes are conflated. This corrupts value and edge-advantage targets by transferring rewards across unrelated decision contexts, producing misleading gradients and premature policy convergence. These results motivate an explicit merge budget in CompGraphRL: limited, reliable merging improves sample efficiency in sparse-reward GUI RL, whereas excessive compression reduces state discriminability and destabilizes learning.

4.5 Ablation Study

Table 3 isolates the two key components of CompGraphRL: graph-based representations and value-function guidance. Replacing graph inputs with trajectories reduces the success rate from 30.4% to 25.3%, showing that graphs capture relational information not available from raw trajectories. Replacing the learned value function with return-to-go yields a similar drop to 25.0%, indicating that value-based advantages are more informative than empirical returns.

Method	Succ. (%) ↑	Δ p.p.
CompGraphRL	$30.4_{\pm 0.2}$	–
w/o Graph	$25.3_{\pm 0.8}$	-5.1
w/o Value Function	$25.0_{\pm 1.7}$	-5.4
w/o Graph & VF	$22.6_{\pm 1.5}$	-7.8
GSPO	$23.8_{\pm 1.2}$	-6.6

Table 3: Ablation of graph representation and value-function guidance on MobileGym with Qwen3.5-9B as the backbone. Δ p.p. denotes the absolute change from CompGraphRL.

Removing both components further reduces performance to 22.6% (a 7.8-point drop). This variant uses trajectory inputs and return-to-go—equivalent to GAE with $\lambda = 1$ and $V = 0$, i.e., the sequence-level advantage in GSPO. Its remaining gap from GSPO (23.8%) is therefore attributable to the CompGraphRL training formulation, whereas the larger gap from the full model underscores the importance of both components. When $\gamma = 1$ and reward shaping is removed, return-to-go becomes a constant terminal reward for all tokens, reducing this variant to GSPO.

5 Conclusion

We present CompGraphRL, a credit-assignment framework for online mobile GUI reinforcement learning using compressed semantic state graphs. It addresses sparse terminal rewards in long-horizon tasks, where equivalent interaction states cannot be reliably matched through exact visual or textual similarity. CompGraphRL encodes states using visual observations, the preceding action, and interaction history; merges functionally equivalent states according to an explicit merge ratio; and propagates credit over the resulting graph via node-value and edge-advantage estimates. This enables terminal rewards to transfer across related trajectories while retaining action-specific differences among visually similar but behaviorally distinct states. Experiments show that CompGraphRL improves both in-domain adaptation and cross-domain generalization over strong RL baselines. Moderate semantic compression is essential: insufficient merging restricts reward sharing, whereas excessive merging reduces state discriminability and yields erroneous credit signals. These results establish semantic graph construction and graph-level value estimation as an effective basis for credit assignment in sparse-reward GUI learning.

AI USE STATEMENT

We have not used generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, clean and reformat datasets, support qualitative and thematic data analysis, interpret results, or implement methods. Generative AI tools were also not used to formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, or propose or refine hypotheses, as these uses are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, edit a research paper to improve readability, draft parts of a research paper, and create or modify scientific figures or images. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by two authors. LLM-generated parts of the paper were independently verified for correctness at least twice. We take responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work strictly adheres to the ICLR Code of Ethics: it involves no human subjects, uses datasets compliant with original licensing agreements (ensuring privacy and legal compliance), and avoids discriminatory biases in experimental design/results; all authors confirm adherence, with no conflicting sponsorships.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of our results. All training and evaluation environments used in this work are public benchmarks: MobileGym, AndroidWorld, and MobileWorld, as described in Section 4.1 and Appendix A.4. The CompGraphRL algorithm, including semantic state compression, controlled graph construction, and graph-based credit assignment, is specified in Section 3; additional notation, merge-rate control details, training hyperparameters, action-space definitions, benchmark protocols, and compression-metric definitions are provided in Appendix A.2, Appendix A.4, and Appendix A.5. Our implementation is built on the publicly available VERL reinforcement-learning framework, with vLLM for rollout generation and FSDP for training, as described in Section 4.1. The backbone models used in our experiments, including Qwen3.5 and GUI-OWL-1.5 variants, are open-weight models available from their original releases. To facilitate independent verification, we provide anonymized source code and benchmark evaluation logs in the supplementary materials.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 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)*, pp. 12248–12267, 2024.

Anthropic. Introducing claude opus 4.8. `https://www.anthropic.com/news/claude-opus-4-8`, May 2026. Accessed: 2026-07-19.

ByteDance Seed. Ui-tars-1.5. `https://seed-tars.com/1.5`, 2025.

Wanxia Cao, Chengzhen Duan, Pei Fu, Pengzhi Gao, Niu Lian, Fazhan Liu, Hui Liu, Heng Qu, Qinzhuo Wu, Zhehao Yu, et al. Xiaomi-gui-0 technical report. *arXiv preprint arXiv:2606.31410*, 2026.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

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

Google. Gemini 3.5 flash. `https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash`, May 2026. Google AI for Developers. Accessed: 2026-07-19.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. Ui-venus technical report: Building high-performance ui agents with rft. *arXiv preprint arXiv:2508.10833*, 2025.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: An efficient RLHF algorithm with robustness to both prompt and reward models. *arXiv preprint arXiv:2501.03262*, 1(3):5, 2025.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and MCP-augmented environments. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6142–6167, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Junbo Li, Peng Zhou, Rui Meng, Meet P Vadera, Lihong Li, and Yang Li. Turn-PPO: Turn-level advantage estimation with PPO for improved multi-turn RL in agentic LLMs. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6227–6243, 2026.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on UI control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154, 2024.

Guangyi Liu, Gao Wu, Congxiao Liu, Pengxiang Zhao, Liang Liu, Mading Li, Qi Zhang, Mengyan Wang, Liang Guo, and Yong Liu. Memgui-agent: An end-to-end long-horizon mobile gui agent with proactive context management. *arXiv preprint arXiv:2606.19926*, 2026a.

Guangyi Liu, Pengxiang Zhao, Gao Wu, Yiwen Yin, Mading Li, Liang Liu, Congxiao Liu, Zhang Qi, Mengyan Wang, Liang Guo, et al. Mobileforge: Annotation-free adaptation for mobile gui agents with hierarchical feedback-guided policy optimization. *arXiv preprint arXiv:2606.19930*, 2026b.

Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. Llm collaboration with multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32150–32158, 2026c.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for GUIs. *arXiv preprint arXiv:2411.00820*, 2024.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *International Conference on Learning Representations*, volume 2026, pp. 129271–129291, 2026d.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22404–22414. IEEE, 2025.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. `https://openai.com/index/gpt-5-6/`, July 2026. Accessed: 2026-07-20.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pp. 406–441, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Bytedance Seed. Seed2. 0 model card: Towards intelligence frontier for real-world complexity. *arXiv preprint arXiv:2607.00248*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Yifan Sui, Xin Huang, Hongbing Li, Fang Xu, Jiahe Lv, Haolong Yan, Yeqing Shen, Litao Liu, Zhimin Fan, Ziyang Meng, et al. Androiddaily: A verifiable benchmark for mobile gui agents on real-world closed-source applications. *arXiv preprint arXiv:2605.27761*, 2026.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. Trace: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026.

Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, et al. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024.

Yunan Wang, Minghui Song, Zihan Zhang, Shaohan Huang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Group-graph policy optimization for long-horizon agentic reinforcement learning. *arXiv preprint arXiv:2606.22995*, 2026.

Taylor Webb, Shanka Subhra Mondal, and Ida Momennejad. A brain-inspired agentic architecture to improve planning with llms. *Nature Communications*, 16(1):8633, 2025.

Dingbang Wu, Rui Hao, Haiyang Wang, Shuzhe Wu, Han Xiao, Zhenghong Li, Bojiang Zhou, Zheng Ju, Zichen Liu, Lue Fan, et al. Mobilegym: A verifiable and highly parallel simulation platform for mobile gui agent research. *arXiv preprint arXiv:2605.26114*, 2026.

Weichu Xie, Haozhe Zhao, Wenpu Liu, Yongfu Zhu, Liang Chen, Minghao Ye, Zirong Chen, Yuqi Xu, Shuai Dong, Ziyue Wang, et al. Step-wise rubric rewards for llm reasoning. *arXiv preprint arXiv:2605.17291*, 2026.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. MobileRL: Online agentic reinforcement learning for mobile GUI agents, 2025. URL https://arxiv.org/abs/2509.18119.

Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, et al. Step-gui technical report. *arXiv preprint arXiv:2512.15431*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025.

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

Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. What's behind ppo's collapse in long-cot? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*, 2025.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, et al. AgentRL: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *arXiv preprint arXiv:2510.04206*, 2025.

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

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025.

Hanzhang Zhou, Panrong Tong, Xu Zhang, Quyu Kong, Chenglin Cai, Tianyu Xia, Gongjie Zhang, Jianan Zhang, Long Li, Long Chen, et al. Qwen-ui-agent technical report: Toward next-generation real-world centric foundation gui agents. *arXiv preprint arXiv:2607.28227*, 2026.

A Appendix

A.1 Method Notation

This section summarizes the notation used by CompGraphRL. The notation follows Section 3.

Table 4: Main notation used in the CompGraphRL method.

Symbol	Meaning
GUI interaction and rollout	
q	User query or task instruction.
o_t	GUI observation, typically a screenshot, received at interaction step t.
a_t	Executable GUI action produced by the policy at step t, such as click, swipe, drag, text input, or navigation.
h_t	Interaction history up to step t, including prior observations, actions, and model responses.
S_t	Decision state used by CompGraphRL at step t; the initial state is $S_0 = (q, o_0)$.
τ_k	The k-th sampled rollout trajectory for a query, with length T_k.
G	Number of rollout trajectories sampled for the same query in group-based online RL.
$R(\cdot)$	Trajectory-level outcome reward.
Semantic state graph construction	
$f_\phi(\cdot)$	Frozen VLM-based semantic state encoder.
z_t	Compressed semantic embedding of state S_t.
$\mathcal{Z}$	Set of all compressed semantic embeddings.
s_{ij}	Cosine similarity between two state or cluster embeddings.
$\sim$	Semantic equivalence relation used for assigning states to the same compressed GUI-state node.
c_i	A compressed GUI-state node.
$\mathcal{C}$	Set of compressed GUI-state nodes.
e_i^j	Directed action-induced transition edge from node c_i to node c_j under action a.
$\mathcal{E}$	Set of directed transition edges.
$\mathcal{T}$	Compressed semantic state-transition graph.
$\mathrm{Succ}(c_i)$	Successor node set of c_i in $\mathcal{T}$.
$\mathrm{Pred}(c_i)$	Predecessor node set of c_i in $\mathcal{T}$.
ρ_{target}	Target merge rate used to set the maximum merge budget.
B_{merge}	Maximum number of accepted non-initial state merges.
q_{ij}	Adjusted merge score for a candidate pair of clusters.
Graph-based credit assignment	
$V(c_i)$	Value estimate of compressed node c_i.
r_i^j	Average immediate reward over transition samples aggregated into edge e_i^j.
γ	Discount factor.
λ	GAE decay parameter.
$\delta(c_i)$	Node-level temporal-difference residual.
$\delta(e_i^j)$	Edge-level temporal-difference residual.
$A(c_i)$	Node-level advantage used for graph-based credit assignment.
$A(e_i^j)$	Edge-level advantage used for graph-based credit assignment.
$G(c_i)$	Target return for training the value model, defined as $V(c_i) + A(c_i)$.

A.2 Dynamic Merge-Rate Control

The controlled graph construction module uses a target merge rate, ρ_{target}, to adapt the compression granularity to each task instance. This design avoids relying on a single global similarity threshold, whose effect can vary substantially across GUI tasks due to task-specific differences in embedding-similarity distributions. Given N observed states and N_{init} trajectory-initial states, the merge budget is defined as

$$B_{\mathrm{merge}} = \mathrm{round}((N - N_{\mathrm{init}})\rho_{\mathrm{target}}) \,.$$

Because every candidate merge must additionally satisfy the quality and structural constraints described below, ρ_{target} specifies an upper bound on the number of accepted merges rather than guaranteeing a particular final merge ratio.

For each candidate pair of state clusters, we combine the cosine similarity between their cluster centers with trajectory-round information. Specifically, we define the adjusted merge score as

$$q_{ij} = \log(\max(s_{ij}, \epsilon)) \left[1 + \frac{\lambda_g}{2} \max(0, \Delta_{ij} - \delta_g) + \frac{\lambda_l}{2} \left(\max(0, \bar{t}_i - \delta_l) + \max(0, \bar{t}_j - \delta_l)\right)\right].$$

Here, s_{ij} denotes the cosine similarity between the centers of clusters i and j; $\bar{t}_i$ and $\bar{t}_j$ denote their average trajectory rounds; $\Delta_{ij} = |\bar{t}_i - \bar{t}_j|$ is the temporal separation between the two clusters; and ϵ is a numerical stability constant. Because $\log(s_{ij}) \leq 0$ for cosine similarities in the relevant range, the multiplicative penalty increases the magnitude of negative scores and thereby reduces the priority of undesirable merges. The first penalty discourages merging clusters that are widely separated in trajectory time, whereas the second penalizes clusters that occur late in an episode. Their respective strengths are controlled by λ_g and λ_l, and their activation thresholds by δ_g and δ_l. Together, these terms reduce merges across distant decision stages and mitigate late-stage semantic aliasing, while preserving beneficial sharing across trajectories.

A.3 Detailed Related Work

GUI Agents. In recent years, GUI Agents have made significant progress, evolving from early models focused on single-step instruction understanding and interface grounding to intelligent agents capable of multi-turn interaction, autonomous planning, and long-horizon navigation. General-purpose foundation models, represented by Opus 4.8 (Anthropic, 2026), GPT-5.6 Sol (OpenAI, 2026), Gemini 3.5 Flash (Google, 2026), Qwen3.8 (Qwen Team, 2026b), Seed2.0 (Seed, 2026), and GLM-5 (Zeng et al., 2026), are increasingly incorporating GUI interaction capabilities into their core competency systems and demonstrating strong performance in complex interface tasks. On this basis, models specifically trained for GUI scenarios can typically further enhance perception, grounding, planning, and execution capabilities.

Existing research primarily improves the performance of GUI Agents across dimensions such as data construction, training paradigms, environment design, and long-horizon reinforcement learning. MobileAgent (Wang et al., 2024), UI-Tars (Qin et al., 2025), and UI-Venus (Gu et al., 2025) have provided an effective data foundation for GUI capability learning through high-quality data filtering, trajectory generation, and iterative trace construction. AutoGLM (Liu et al., 2024), MAI-UI (Zhou et al., 2025), Step-GUI (Yan et al., 2025), Xiaomi-GUI-0 (Cao et al., 2026), and UI-Venus-1.5 (Venus Team et al., 2026) leverage stable and scalable interaction environments to collect high-quality GUI trajectories, and combine training workflows such as Supervised Fine-Tuning (SFT), Action-level Reinforcement Learning (Action-level RL), and Online Reinforcement Learning (Online RL) to gradually improve end-to-end GUI navigation capabilities. MobileAgent-v3 (Ye et al., 2025) and MobileAgent-v3.5 (Xu et al., 2026) further employ Continual Pre-Training (CPT) to enhance the models' perception, grounding, planning, and execution capabilities, while open-sourcing the GUI-OWL series of models and further improving end-to-end interaction performance through multi-agent orchestration systems. In addition, UI-Tars-2 (Wang et al., 2025), mobileRL (Xu et al., 2025), and AgentRL (Zhang et al., 2025) focus on utilizing online reinforcement learning to enhance the models' decision-making capabilities in multi-turn, long-horizon tasks, and improve data collection efficiency through asynchronous training and sampling mechanisms. MemGUI-Agent (Liu et al., 2026a) strengthens the model's long-term memory capabilities through a memory compression mechanism. Qwen-UI-Agent (Zhou et al., 2026), oriented towards real interaction scenarios, has constructed a diverse sandbox environment covering mobile, computer operation, web interaction, and DeepSearch tasks, and combined with AutoResearch-style data flywheels and online reinforcement learning, has achieved excellent end-to-end performance.

With the continuous improvement of GUI Agent capabilities, mobile GUI benchmarks have also evolved from offline static datasets to dynamic interactive environments. Early work was primarily based on offline static datasets such as AndroidControl (Li et al., 2024) and GUI Odyssey (Lu et al., 2025); subsequently, simulator-based benchmarks such as AndroidWorld (Rawles et al., 2025) and MobileWorld (Kong et al., 2026) further supported interactive evaluation; furthermore, AndroidDaily (Sui et al., 2026) extended evaluation to real device environments. To improve environment fidelity and reduce the cost of large-scale sampling, MobileGym (Wu et al., 2026) proposed a lightweight, fully controllable simulation platform based on browser hosting. Benefiting from

the open-source implementation of MobileGym, the method in this paper can conduct large-scale rollouts with lower resource overhead, avoiding the massive computational resources required by environments such as AndroidWorld or MobileWorld during the sampling phase.

Reinforcement Learning for Long-horizon Agents.

Reinforcement learning (RL) has emerged as a key paradigm for improving the adaptability and decision-making capabilities of large language model (LLM) agents in dynamic, open-ended environments. Recent approaches, including GRPO (Shao et al., 2024), PPO (Schulman et al., 2017), Reinforce++ (Hu et al., 2025), RLOO (Ahmadian et al., 2024), DAPO (Yu et al., 2026), have demonstrated strong performance in large-scale reinforcement learning with verifiable rewards (RLVR). GSPO (Zheng et al., 2025) estimates sequence-level advantages from grouped trajectories, which improves optimization stability over GRPO on long-horizon, multi-turn tasks. VC-PPO (Yuan et al., 2025) attributes the collapse of critic-based PPO on long-horizon reasoning to value-function optimization, showing that a properly optimized (typically pretrained) critic is essential for stable value-based credit assignment. However, conventional outcome-based rewards provide supervision only upon task completion. While effective for short-horizon reasoning, they become highly sparse in long-horizon settings involving dozens or even hundreds of tool calls. Moreover, estimating action advantages from returns over complete trajectories can introduce substantial variance. These limitations have motivated recent work on fine-grained credit assignment for multi-turn interaction and long-horizon optimization. PRIME (Cui et al., 2025) introduces process-level reinforcement learning with an implicit process reward model (PRM) that derives dense token-level process rewards from outcome-only labels, thereby mitigating reward sparsity and improving fine-grained credit assignment in long-horizon reasoning without costly step-level supervision. MT-GRPO (Zeng et al., 2025) introduces turn-level verifiable rewards and demonstrates the benefit of turn-level credit assignment. TRACE (Tao et al., 2026) is a turn-level credit-assignment approach that estimates the contribution of each interaction turn and assigns turn-specific rewards, providing finer-grained learning signals for long-horizon agents than trajectory-level outcome rewards. iStar (Liu et al., 2026d) introduces an actor-critic framework that derives implicit step-level rewards from outcome-only labels by training a critic with a cross-entropy loss, thereby providing dense credit assignment for multi-turn agentic tasks without explicit step-level supervision. GiGPO (Feng et al., 2026) evaluates candidate actions within local states to capture local decision preferences, but struggles to establish meaningful comparisons across non-repeated states. Nonetheless, Turn-PPO (Li et al., 2026) elevates the modeling granularity of the Markov decision process (MDP) from the token level to the turn level, treating the input and output of each interaction turn as a complete state-action pair. Most closely related to our work, G2PO (Wang et al., 2026) explicitly reconstructs sampled trajectories as state-transition graphs and reduces sampling variance and trajectory-dependency bias through group-aggregated state-value estimation. Our work builds on the insight that graph structure is useful for long-horizon credit assignment, but targets a complementary bottleneck in mobile GUI agents. Whereas G2PO can group states when repeated observations or environment-derived state identities are available, mobile GUI interaction requires semantic state abstraction under noisy visual observations, coordinate perturbations, layout variations, device-specific rendering, and history-dependent meanings. CompGraphRL therefore focuses on constructing reliable semantic state graphs with VLM-based compression, merge-budget control, and temporal constraints, and then applies graph-structured credit propagation on top of this compressed GUI state graph.

A.4 Implementation Details

A.4.1 Baseline Implementations

We compare against three online RL baselines under a controlled protocol. **GSPO** (Zheng et al., 2025) assigns a sequence-level advantage to each sampled trajectory and serves as the stable outcome-based baseline. **GiGPO** (Feng et al., 2026) forms inner groups from decisions associated with the same local state and contrasts candidate actions within each group, providing finer-grained supervision when states recur across rollouts. **G2PO** (Wang et al., 2026) reconstructs sampled trajectories as a state-transition graph and estimates group-aggregated state values to reduce trajectory-dependent variance. In their original settings, GiGPO and G2PO rely on repeated states or environment-derived state identities. Since exact state repetitions are virtually absent in GUI trajectories, we adapt both methods by treating two states as identical when the similarity between their observation images exceeds the matching threshold. They do not use VLM-based semantic em-

beddings, action or history information, controlled semantic merging, or the CompGraphRL value model.

For a fair method-level comparison, all baselines use the same Qwen3.5 initialization checkpoints, MobileGym training split, eight trajectories per query, rollout prompts, action space, reward shaping, learning rate, asynchronous training framework, checkpoint-selection rule, and three evaluation seeds described in Section 4.1. GiGPO and G2PO are evaluated with both Qwen3.5-9B and Qwen3.5-27B and transferred without further adaptation to AndroidWorld and MobileWorld, matching the evaluation protocol for CompGraphRL. Thus, the comparisons in Table 1 primarily differ in how trajectory outcomes are converted into policy-optimization credit.

A.4.2 Training Hyperparameters

We use Qwen3.5-9B as the initialization checkpoint for the GUI policy and train it through online interaction with a mobile environment. Rollout generation and policy optimization are performed asynchronously on a single eight-GPU node: four GPUs are allocated to vLLM rollout workers, and the remaining four GPUs are used by the learner. For each instruction, the policy samples eight trajectories, each comprising at most 50 user turns and 50 assistant turns. At each turn, the model receives up to three screenshots, with spatial references represented in absolute pixel coordinates.

The policy is optimized with a learning rate of 5×10^{-7}, token-mean loss aggregation, asymmetric clipping bounds, and an explicit KL-regularization term. The asynchronous learner synchronizes its parameters with the rollout workers every two update steps and accepts rollouts generated by policies that are at most two versions stale. This design overlaps environment interaction with policy optimization while constraining the distribution shift between the rollout and learner policies.

The value model is pretrained on a subset of the collected trajectories and subsequently frozen throughout policy optimization. It is trained separately using a frozen Qwen3-VL-4B-Instruct backbone, with optimization restricted to the value-prediction head. We use Adam with a cosine learning-rate schedule and train the value head for three epochs. Following standard PPO, the value model regresses each aggregated node value toward its target return. For an aggregated node c_i, the target return is defined as

$$G(c_i) = V(c_i) + A(c_i).$$

The value model is trained by minimizing the mean squared error

$$\mathcal{L}_{\text{value}} = \frac{1}{|\mathcal{C}|} \sum_{c_i \in \mathcal{C}} \left(V(c_i) - G(c_i)\right)^2,$$

where $\mathcal{C}$ denotes the set of aggregated nodes. The resulting node-value estimates are used for graph-based credit propagation, with discount factor $\gamma = 0.95$ and GAE parameter $\lambda = 0.99$.

For GUI-specific reward shaping, each action at interaction step t receives the following reward:

$$r_t = \begin{cases} +4, & \text{if the action successfully completes the task,} \\ -2, & \text{if the task terminates unsuccessfully,} \\ -0.1, & \text{if the action is invalid or repetitive,} \\ -4, & \text{if the generated action is malformed,} \\ 0, & \text{otherwise.} \end{cases}$$

This reward design provides immediate feedback for GUI-specific failure modes while preserving zero reward for valid, non-terminal interactions.

At each interaction turn, the policy emits a single structured action. For spatial actions, the model outputs normalized coordinates in the range $[0, 1000]$, which are subsequently mapped to absolute pixel coordinates based on the device's screen resolution. The coordinate argument of `swipe` is optional and is used only when the gesture must be anchored to a particular UI element. Table 5

summarizes the training configuration, GUI-specific reward shaping scheme, and complete action space.

Table 5: Training hyperparameters, and structured action space used in our experiments.

Category	Setting	Value / Description
Rollout	Rollout engine	vLLM V1
	Rollout execution	Fully asynchronous
	Trajectories per prompt	8
	Maximum prompt length	10,240 tokens
	Maximum response length	1,024 tokens
	Maximum context length	32,768 tokens
	Maximum user / assistant turns	50 / 50
	Maximum images per turn	3
	Maximum pixels per image	1,036,800
	Sampling temperature	1.0
	Sampling top-p	1.0
	Sampling top-k	-1
	Tensor-parallel size	4
	GPU-memory utilization	0.6
	Chunked prefill	Enabled
Policy optimization	Learning rate	5×10^{-7}
	Warmup steps	0
	Weight decay	0.1
	Batch size	16
	Loss aggregation	Mean over generated tokens
	Lower / upper clipping ratio	3×10^{-4} / 4×10^{-4}
	KL regularization coefficient	1×10^{-3}
	KL penalty in environment reward	Disabled
	Entropy coefficient	1×10^{-3}
	Gradient clipping	1.0
	Overlong-response buffer	1,024 tokens
	Overlong penalty factor	1.0
Async training	Maximum rollout staleness	2
	GPUs for rollout / optimization	4 / 4
	Distributed strategy	FSDP2
	Gradient checkpointing	Enabled
Value model	Backbone	Qwen3-VL-4B-Instruct
	Adam β_1	0.9
	Adam β_2	0.999
	Adam ϵ	10^{-5}
	Learning rate	10^{-4}
	Learning-rate schedule	Cosine
	Batch size	256
	Training epochs	3
	Maximum sequence length	512 tokens
	Discount factor γ	0.95
	GAE parameter λ	0.99
Merge-rate control	Temporal-span penalty strength λ_g	0.8
	Late-episode penalty strength λ_l	0.4
	Temporal-span threshold δ_g	8
	Late-episode threshold δ_l	16

Continued on next page

Table 5 – continued

Category	Setting	Value / Description
Reward shaping	Successful termination	$+4$
	Failed termination	-2
	Invalid or repeated action	-0.1
	Malformed action output	-4
	Valid non-terminal action	0
Action space	**click** *coordinate*: $[x, y]$	Click the specified UI location.
	long_press *coordinate*: $[x, y]$	Press and hold the specified UI location.
	double_click *coordinate*: $[x, y]$	Double-click the specified UI location.
	type *text*: string	Enter text in the active input field.
	swipe *direction*: up/down/left/right; *coordinate*: $[x, y]$ (optional)	Swipe in the specified direction.
	open *text*: application name	Launch the specified application.
	drag *start/end coordinates*: $[x_1, y_1]/[x_2, y_2]$	Drag between two UI locations.
	system_button *button*: back/home/menu/enter	Trigger a system-level button.
	wait *No arguments*	Wait for the interface to update.
	terminate *status*: success/fail	Terminate with the predicted status.
	answer *text*: answer string	Return a textual answer.

A.4.3 Evaluation Benchmarks

MobileGym (Wu et al., 2026) is an open-source, browser-based mobile simulation benchmark for scalable and reproducible training and evaluation of mobile GUI agents. The environment state is fully programmable and includes 28 simulated applications and 416 parameterized task templates. Each task is paired with a deterministic code-based evaluator with sub-millisecond execution latency, enabling reliable and fine-grained assessment of agent behavior. MobileGym supports large-scale deployment, with a single server capable of running up to 256 parallel instances. Each instance requires approximately 400 MB of memory and has a cold-start latency of around 3 seconds. The benchmark provides 160 training tasks and 256 evaluation tasks spanning diverse mobile interaction scenarios.

In this work, we integrate MobileGym into the Verl reinforcement learning training pipeline to support interactive RL for mobile GUI agents. We construct task instances using fixed parameters or runtime environment-time placeholders to ensure consistency among task prompts, simulator setups, answer sheets, and judge inputs. During rollout, model-generated actions, including taps, text input, swipes, and drag operations, are executed in MobileGym through an action executor. MobileGym's native ORM-based evaluator and judge return structured feedback, including `success`, `progress`, `issues`, and `warnings` in `judge_details`. Our training reward is primarily based on the `success` signal. We further leverage MobileGym's multi-device and multi-session scheduling, multithreaded execution, and high-load capability to support stable environment interaction and batched evaluation under large-scale asynchronous rollouts. After completing both online and offline evaluation on MobileGym, we further evaluate the performance of the trained models on AndroidWorld and MobileWorld. This integration enables MobileGym to serve not only as a

comprehensive benchmark but also as a scalable environment for end-to-end reinforcement learning of mobile GUI agents.

AndroidWorld (Rawles et al., 2025) is a widely adopted benchmark for evaluating autonomous mobile agents, developed by Google DeepMind. It provides a reproducible and deterministic evaluation environment built on real Android emulators, supporting reliable assessment of agents in realistic mobile interaction settings. The benchmark comprises 116 manually designed tasks across 20 real Android applications. To mitigate overfitting to a fixed test set, AndroidWorld dynamically instantiates tasks with randomized parameters, yielding millions of task variations. Each task is specified through a parameterized natural-language instruction and is accompanied by dedicated initialization, success-checking, and cleanup logic, ensuring consistent and repeatable evaluation. Owing to its realistic application coverage, rigorous task design, and deterministic evaluation protocol, AndroidWorld serves as an authoritative benchmark for measuring the generalization and real-world usability of mobile GUI agents.

To further evaluate model performance in more realistic and challenging mobile interaction scenarios, we additionally evaluate our model on MobileWorld (Kong et al., 2026). MobileWorld is a comprehensive benchmark comprising 201 tasks across 20 commonly used mobile applications. Its tasks require an average of 27.8 steps, nearly twice that of AndroidWorld, and 62.2% involve cross-app interactions, placing greater demands on long-horizon planning and information integration. MobileWorld also includes agent–user interaction tasks for intent clarification and MCP-enhanced tasks that combine GUI navigation with external tool use. By emulating production-grade applications with open-source alternatives and providing deterministic end-to-end evaluation, MobileWorld offers a rigorous and realistic benchmark for assessing the robustness and practical effectiveness of mobile GUI agents.

A.5 Compression Pattern and Metric Definitions

This appendix provides the experimental details omitted from the compression analysis in Section 4.3. The goal of this analysis is to isolate the effect of state compression from the downstream policy optimizer: all compression strategies are evaluated on the same pool of MobileGym rollouts, use the same state-transition logging schema, and are compared against the same human state partition. The only changed component is how candidate GUI states are mapped to compressed graph nodes.

Rollout pool and annotation protocol. We sample a fixed set of 30 MobileGym tasks and collect $K = 8$ trajectories for each task under the same backbone policy and action-space adapter used in the main experiments, yielding 240 annotated trajectories in total. Following the experimental design, the diagnostic pool is constructed to cover short-horizon and long-horizon tasks, form filling, search, cross-page navigation, setting modification, backtracking or undo operations, loop-like browsing behavior, and visually similar states with different task semantics. Each interaction step records `task_id`, `episode_id`, `step_id`, screenshot, visible text, previous action, action arguments, reward, terminal flag, success flag, task-progress metadata, and policy-side identifiers. Human annotators then group states that are functionally equivalent for the next GUI decision. Two states are annotated as equivalent only when they present the same task-relevant affordances, require the same decision under the current user instruction, and differ only in nuisance factors such as rendering details, coordinates, minor dynamic content, or repeated visits to the same functional screen. States are kept separate when they correspond to different task stages, different filled slots, different error dialogs, different hidden app states, or the same visual screen reached after semantically different histories. Each task is independently annotated by two annotators using only the candidate-state equivalence relation and the resulting semantic state identifiers. Disagreements are restricted to the affected candidate states and resolved by a third adjudicator who inspects the full trajectory context; the final adjudicated partition is used as the gold partition for ARI, NMI, VI_{norm}, homogeneity, completeness, and pairwise merge metrics. State-transition edges are then derived automatically from rollout order and are not separately hand-labeled.

Compared compression patterns. Let $S_t = (a_{t-1}, o_t, h_{t-1})$ denote the state used by CompGraphRL. All methods first build candidate states from the same logged trajectories and then construct a directed graph by adding an edge for each observed action-induced transition between the resulting clusters. We compare six compression patterns.

- **Linear Trajectory Chain (LTC)** keeps each trajectory as an independent chain and only shares the initial task node across rollouts. This setting approximates no semantic merging and tests whether cross-trajectory state reuse is necessary.
- **Image Similarity Clustering (ISC)** represents each state using only the current screenshot or visual embedding and merges states whose visual similarity exceeds the common threshold. It captures layout-level repetition but ignores action history and task progress.
- **Image-Action Clustering (IAC)** augments ISC with the previous action category, so two visually similar states are merged only when their preceding action types are also compatible. This variant tests whether a shallow action-conditioned visual representation is sufficient.
- **Textual State Compression (TSC)** converts each GUI state into a textual description derived from visible text, UI-tree content, and state metadata, then computes state similarity using Qwen3-4B-Embedding. It evaluates the text-first alternative to VLM compression.
- **Round-aware Textual Compression (RTC)** extends TSC with interaction-round information, which discourages merging states that appear at very different trajectory stages.
- **VLM State Compression (VSC)** is the CompGraphRL compressor used in the main method. It encodes screenshot-level visual evidence, UI semantics, the previous action, and compact interaction history into the state representation $z_t = f_\phi(S_t)$, and then applies the controlled merge procedure described in Section 3.2.

For the method-level comparison in Table 2, all similarity-based methods use the same merge threshold, 0.95, so that differences mainly reflect the information available to the state representation rather than threshold tuning.

Metric aggregation protocol. All compression metrics in Table 2, including the partition-level and pairwise metrics defined below, are computed separately for each of the 30 tasks and then macro-averaged across tasks. Thus, every task receives equal weight regardless of its number of states or state pairs. In particular, the reported pairwise precision, recall, and F1 are not obtained from a single confusion matrix pooled over all tasks; pooling TP, FP, and FN across tasks will therefore not reproduce the table values.

Partition-level metrics. For each task, let $\mathcal{P}$ be the human state partition and $\widehat{\mathcal{P}}$ be an automatic partition. We report five partition-level metrics.

- **Adjusted Rand Index (ARI)** measures pairwise agreement between $\mathcal{P}$ and $\widehat{\mathcal{P}}$ after correcting for chance. Given the contingency table $\{n_{ij}\}$ between the two partitions, with row sums a_i and column sums b_j, it is defined as

$$\mathrm{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\frac{1}{2}\left(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right) - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}}.$$

- **Normalized Mutual Information (NMI)** measures the normalized information overlap between the human and automatic partitions:

$$\mathrm{NMI}(\mathcal{P}, \widehat{\mathcal{P}}) = \frac{2I(\mathcal{P}; \widehat{\mathcal{P}})}{H(\mathcal{P}) + H(\widehat{\mathcal{P}})},$$

  where $I(\cdot;\cdot)$ is mutual information and $H(\cdot)$ is entropy.
- **Variation of Information (VI)** measures the distance between two partitions:

$$\mathrm{VI}(\mathcal{P}, \widehat{\mathcal{P}}) = H(\mathcal{P}) + H(\widehat{\mathcal{P}}) - 2I(\mathcal{P}; \widehat{\mathcal{P}}).$$

  For a task with n candidate states, we report the log-cardinality-normalized score

$$VI_{\mathrm{norm}}(\mathcal{P}, \widehat{\mathcal{P}}) = \frac{\mathrm{VI}(\mathcal{P}, \widehat{\mathcal{P}})}{\log n},$$

using the same logarithm base as the entropy terms. This normalization makes VI comparable across tasks with different numbers of annotated states; $VI_{\mathrm{norm}} = 0$ indicates identical partitions, and lower values are better.

- **Homogeneity** is high when each predicted cluster contains states from a single human cluster, and therefore penalizes over-merging. It is defined as

$$\mathrm{Homogeneity} = 1 - \frac{H(\mathcal{P} \mid \widehat{\mathcal{P}})}{H(\mathcal{P})}.$$

- **Completeness** is high when all states from the same human cluster are assigned to the same predicted cluster, and therefore penalizes over-splitting. It is defined as

$$\mathrm{Completeness} = 1 - \frac{H(\widehat{\mathcal{P}} \mid \mathcal{P})}{H(\widehat{\mathcal{P}})}.$$

These metrics are complementary: a method can obtain high homogeneity by refusing to merge many equivalent states, or high completeness by merging too aggressively, so the combined metrics are needed to distinguish conservative and aggressive compression errors.

Pairwise merge metrics. We also evaluate each unordered pair of states as a binary merge decision. A true positive is a pair that is merged by the automatic method and belongs to the same human-annotated state group. A false positive is an automatically merged pair that human annotation keeps separate. A false negative is a pair that belongs to the same human group but is not merged by the automatic method. Pairwise precision, recall, and F1 are then defined in the standard way:

- **Precision** measures the fraction of automatically merged pairs that are also merged in the human partition:

$$\mathrm{Precision} = \frac{TP}{TP + FP}.$$

- **Recall** measures the fraction of human-equivalent pairs that are recovered by the automatic partition:

$$\mathrm{Recall} = \frac{TP}{TP + FN}.$$

- **F1-score** is the harmonic mean of pairwise precision and recall:

$$\mathrm{F1} = \frac{2\,\mathrm{Precision}\,\mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}.$$

Pairwise precision is most sensitive to harmful over-merging, while pairwise recall is most sensitive to missed sharing opportunities. We therefore use pairwise F1 as a compact indicator of whether a compression method balances the two failure modes.

Summary. The compression strategy analysis relies on a fixed set of compression patterns, a controlled merge-ratio protocol, and explicit evaluation metrics. The method definitions clarify what information each state representation uses, while the partition-level and pairwise metrics define how automatic GUI-state compression is compared against the human state partition. Under this protocol, VSC directly corresponds to the semantic state compressor used in CompGraphRL: by jointly encoding visual evidence, UI semantics, previous actions, and compact interaction history, it targets the functional GUI-state equivalence that simpler image-only, action-conditioned, or text-first variants cannot fully capture. These definitions therefore make the compression analysis reproducible and explain how the measured compression quality supports the effectiveness of our VSC module before downstream policy optimization is considered.

A.6 Prompt Templates

Here, we provide the prompts on MobileGym used to guide the planning and action processes of GUI agents built on the Qwen3.5 and GUI-OWL-1.5 backbones.

```
You are a GUI agent. You are given a task and your action history, with
    screenshots. You need to perform the next action to complete the task
    .

## Output Format
For each function call, return the thinking process in <think> </think>
    tags, and a json object with function name and arguments within <
    tool_call></tool_call> XML tags:
```
<think>
...
</think>
<tool_call>
{"name": "mobile_use", "arguments": <args-json-object>}
</tool_call>
```

## Action Space

{"action": "click", "coordinate": [x, y]}
{"action": "long_press", "coordinate": [x, y]}
{"action": "double_click", "coordinate": [x, y]}
{"action": "type", "text": ""}
{"action": "swipe", "direction": "up or down or left or right", "
    coordinate": [x, y]} # "coordinate" is optional. Use the "coordinate"
     if you want to swipe a specific UI element.
{"action": "open", "text": "app_name"}
{"action": "drag", "start_coordinate": [x1, y1], "end_coordinate": [x2,
    y2]}
{"action": "system_button", "button": "button_name"} # Options: back,
    home, menu, enter
{"action": "wait"}
{"action": "terminate", "status": "success or fail"}
{"action": "answer", "text": "xxx"} # Use escape characters \\', \\", and
     \\n in text part to ensure we can parse the text in normal python
    string format.


## Note
- Write a small plan and finally summarize your next action (with its
    target element) in one sentence in <think></think> part.
- Available Apps: `["WeChat", "Alipay", "Maps", "Calendar", "Clock", "
    Notes", "Settings", "Phone", "Messages", "Gallery", "Files", "Browser
    ", "Calculator", "Weather", "Compass", "Bilibili", "RedNote", "WeRead
    ", "Tencent Meeting", "Railway 12306", "Spotify", "Reddit", "X", "
    eBay", "answerSheet"]`.
You should use the `open` action to open the app as possible as you can,
    because it is the fast way to open the app.
- You must follow the Action Space strictly, and return the correct json
    object within <think> </think> and <tool_call></tool_call> XML tags.
```

Prompt Template for Qwen3.5

```
# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags
    :
<tools>
{"type": "function", "function": {"name_for_human": "mobile_use", "name":
     "mobile_use", "description": "Use a touchscreen to interact with a
    mobile device, and take screenshots.
```

```
* This is an interface to a mobile device with touchscreen. You can perform actions like clicking, typing, swiping, etc.
* Some applications may take time to start or process actions, so you may need to wait and take successive screenshots to see the results of your actions.
* The screen's resolution is 1000x1000.
* Make sure to click any buttons, links, icons, etc with the cursor tip in the center of the element. Don't click boxes on their edges unless asked.", "parameters": {"properties": {"action": {"description": "
  The action to perform. The available actions are:
* `key`: Perform a key event on the mobile device.
  - This supports adb's `keyevent` syntax.
  - Examples: \"volume_up\", \"volume_down\", \"power\", \"camera\", \"clear\".
* `click`: Click the point on the screen with coordinate (x, y).
* `long_press`: Press the point on the screen with coordinate (x, y) for specified seconds.
* `swipe`: Swipe from the starting point with coordinate (x, y) to the end point with coordinates2 (x2, y2).
* `type`: Input the specified text into the activated input box.
* `system_button`: Press the system button.
* `open`: Open an app on the device.
* `wait`: Wait specified seconds for the change to happen.
* `answer`: Terminate the current task and output the answer.
* `terminate`: Terminate the current task and report its completion status.", "enum": ["key", "click", "long_press", "swipe", "type", "system_button", "open", "wait", "answer", "terminate"], "type": "string"}, "coordinate": {"description": "(x, y): The x (pixels from the left edge) and y (pixels from the top edge) coordinates to move the mouse to. Required only by `action=click`, `action=long_press`, and `action=swipe`.", "type": "array"}, "coordinate2": {"description": "(x, y): The x (pixels from the left edge) and y (pixels from the top edge) coordinates to move the mouse to. Required only by `action=swipe`.", "type": "array"}, "text": {"description": "Required only by `action=key`, `action=type`, `action=open`, `action=answer`.", "type": "string"}, "time": {"description": "The seconds to wait. Required only by `action=long_press` and `action=wait`.", "type": "number"}, "button": {"description": "Back means returning to the previous interface, Home means returning to the desktop, Menu means opening the application background menu, and Enter means pressing the enter. Required only by `action=system_button`", "enum": ["Back", "Home", "Menu", "Enter"], "type": "string"}, "status": {"description": "The status of the task. Required only by `action=terminate`.", "type": "string", "enum": ["success", "failure"]}}, "required": ["action"], "type": "object"}, "args_format": "Format the arguments as a JSON object."}}
</tools>

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>

# Response format

Response format for every step:
1) Action: a short imperative describing what to do in the UI.
2) A single <tool_call>...</tool_call> block containing only the JSON: {"name": <function-name>, "arguments": <args-json-object>}.

Rules:
- Output exactly in the order: Action, <tool_call>.
- Be brief: one for Action.
- Do not output anything else outside those two parts.
```

```
- If finishing, use action=terminate in the tool call.
```

Prompt Template for GUI-OWL-1.5

For AndroidWorld and MobileWorld evaluation, we use the same prompt templates as above and only modify the `Available Apps` field to match the applications provided by the corresponding benchmark environment.

A.7 UI Analysis



Figure 4: Functionally identical GUI pages with varying dynamic content.

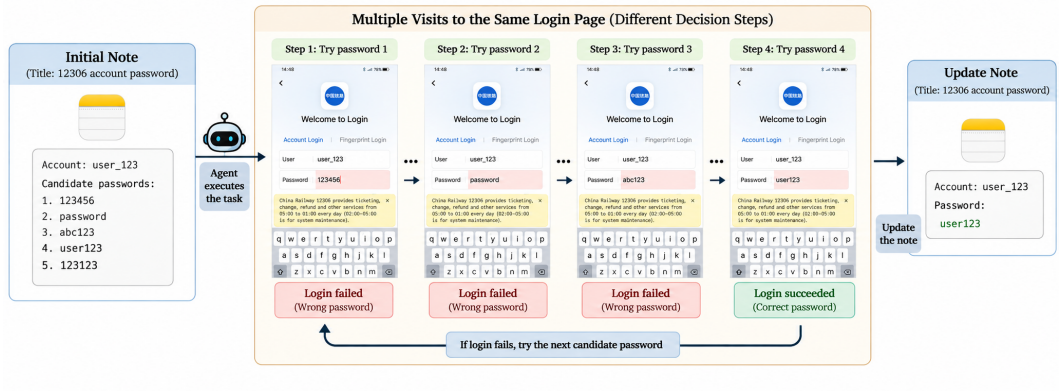


Figure 5: The same visual representation depicts various phases within the decision-making cycle.

Figure 4 illustrates a class of functionally equivalent GUI states that differ substantially in visual appearance due to variations in layout coordinates, screen resolution, themes, and dynamic content. In the relatively stable MobileGym environment, standard rollouts typically produce limited visual variation. To better emulate the thematic diversity encountered on real-world mobile devices, we introduce controlled perturbations by modifying font attributes. As highlighted by the dashed rectangles, rendering text with full-width characters substantially increases the spatial extent of numerals, symbols, and alphabetic characters. Despite these pronounced visual differences, the resulting pages preserve identical functional logic and should therefore induce the same successor states. Consistent with this expectation, during the task of "awarding a coin to a specific Bilibili video without liking it," our method correctly groups these visually heterogeneous observations into a single cluster.

As illustrated in Figure 5, GUI-agent tasks also present the complementary challenge that visually identical states may correspond to different stages of decision making, a phenomenon commonly referred to as *perceptual aliasing*. Consider a workflow in which an agent retrieves candidate passwords for an account (e.g., `user_123`) from a memo, verifies them, and subsequently updates the memo to retain only the correct credential. The agent may visit the same memo interface multiple times during this process. Although these observations are pixel-wise identical, they serve fundamentally different purposes: the initial visit supports *information retrieval*, whereas the subsequent visit enables *content modification* conditioned on the verification result. Consequently, clustering states solely according to raw visual similarity is insufficient and may be misleading. Such a coarse clustering strategy disregards essential *temporal context* and *task semantics*, collapsing functionally distinct states into a shared representation and impairing the agent's ability to track task progress.

A.8 Trajectories Analysis

Figures 6 and 7 compare the proposed method with Qwen3.5-9B on the same GUI task: *adjusting the Alipay font size to one level below the maximum setting*. The results indicate that our method substantially improves task-completion robustness, error recovery, and the execution of complex interaction operations.

When Qwen3.5-9B executes an incorrect action and reaches an intermediate UI state that deviates from the intended target, it often fails to recognize the discrepancy and revise its subsequent decisions accordingly. As a result, the model may repeatedly perform similar actions, become trapped in an interaction loop, and ultimately fail to complete the task. In contrast, our method continuously evaluates intermediate execution outcomes. When the observed UI state differs from the expected state, it promptly updates its action plan and exploration trajectory, identifies the appropriate function entry, and completes the remaining configuration steps.

In particular, *swipe* and *drag* are challenging interaction primitives in GUI automation because their success depends on accurate selection of the start and end coordinates, direction, displacement, and interpretation of UI feedback. Qwen3.5-9B is susceptible to errors in these operations, which can lead to unintended interface states or execution failures. By contrast, our method improves the reliability of *swipe* and *drag* through more effective action selection and execution-feedback mechanisms, thereby increasing the success rate on complex GUI tasks.

Figure 8 further illustrates the performance of our method on long-horizon, cross-application tasks. Even when a task requires switching among multiple applications and executing a lengthy sequence of actions, the model maintains strong task-completion quality and execution stability. Under a predefined action budget, it can detect deviations and erroneous actions during execution and apply corrective actions in a timely manner. This capability prevents errors from propagating to subsequent steps, thereby improving the overall success rate and robustness of long-horizon tasks.

Figure 9 demonstrates the advantages of the proposed method on tasks that require complex computation and reasoning. Consider the composite task of "calculating the total duration of meetings on February 4th and navigating to the Answer Sheet app to submit the result." The model effectively decomposes the task, retrieves relevant meeting records from the daily schedule, extracts temporal information, and accumulates discrete duration values through multi-turn reasoning over the in-context interaction history. It then transitions to the target application and completes the required submission procedure. This example highlights the reliability of our method for long-horizon task execution and precise numerical reasoning.

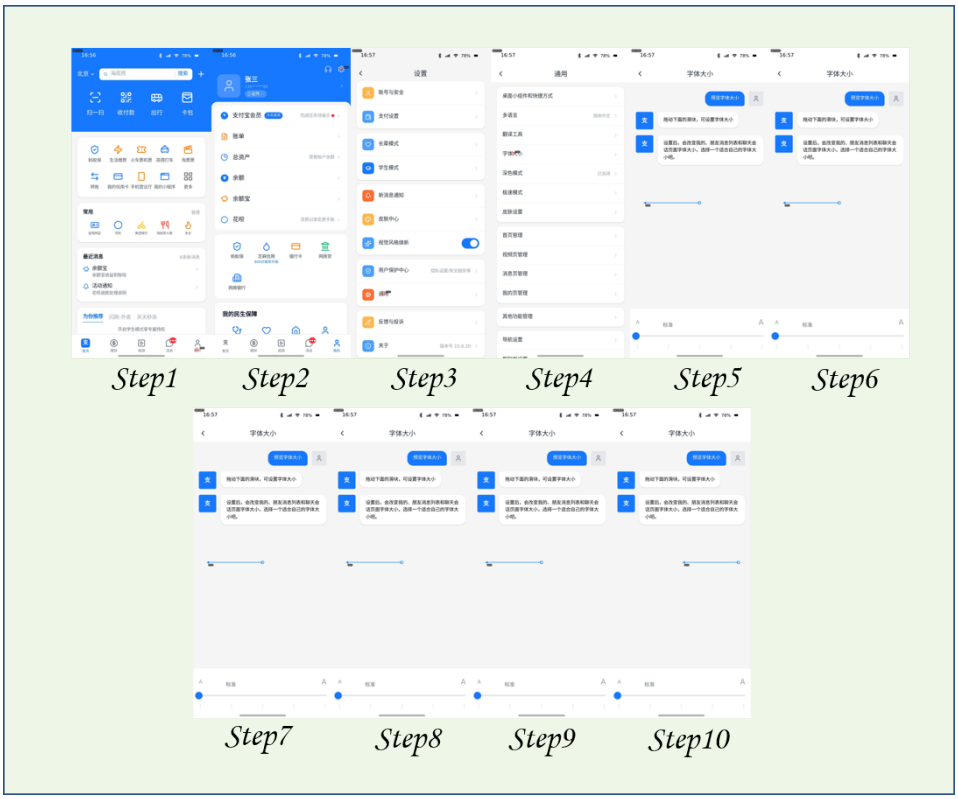


Figure 6: A trajectory that failed on the Qwen3.5-9B base model

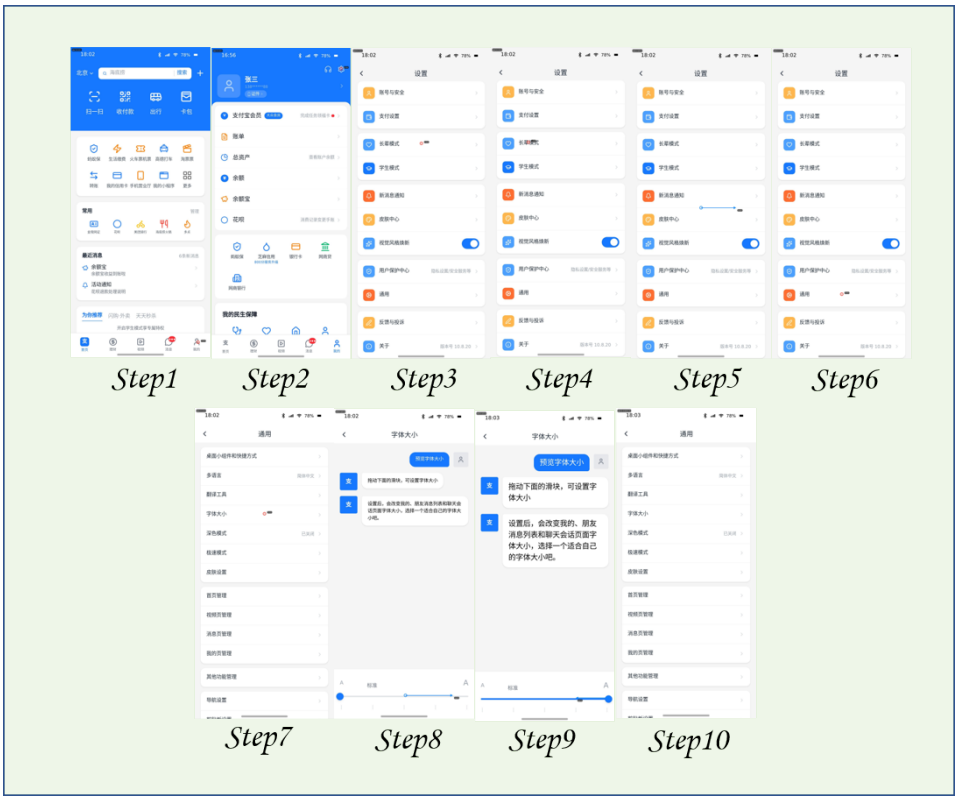


Figure 7: A trajectory that succeeded in CompGraphRL

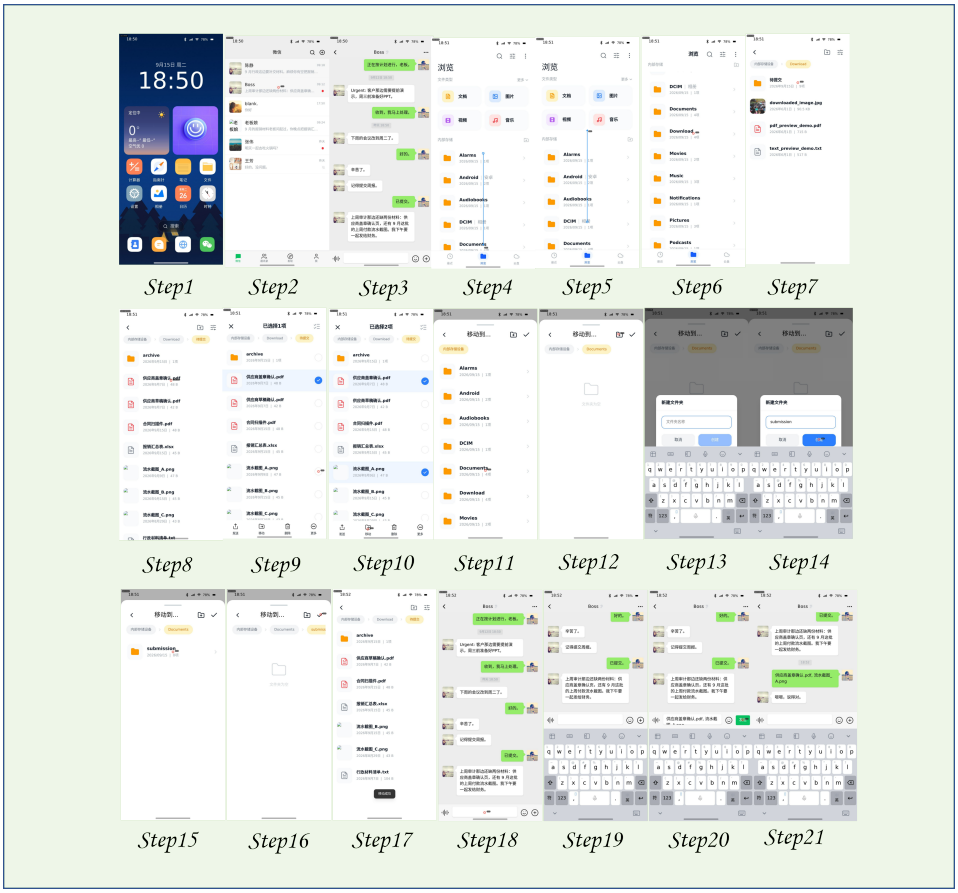


Figure 8: The success trajectory of CompGraphRL in a long-horizon cross-app task

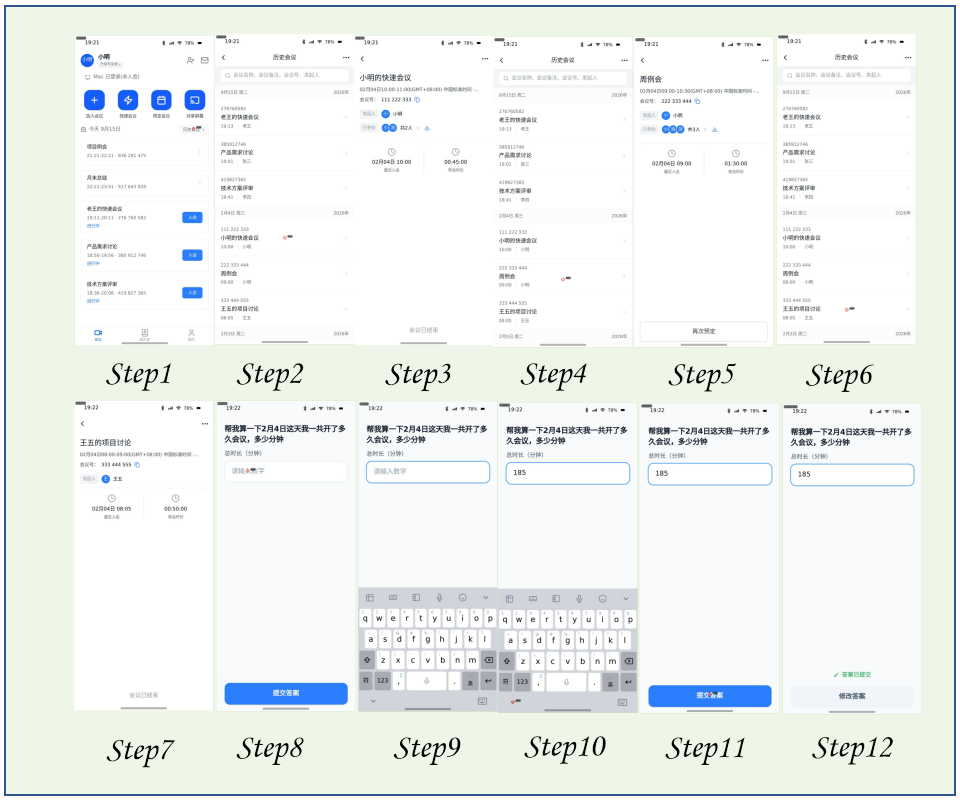


Figure 9: The success trajectory of CompGraphRL in a computational task