# JHiMA-WebRL: A Joint Hierarchical Multi-Agent Reinforcement Learning Framework for Long-Horizon Web Tasks

**Anonymous authors**
Paper under double-blind review

## Abstract

Multimodal web agents have shown promising capabilities for interacting with websites through graphical user interfaces (GUIs), yet realistic web tasks often require long-horizon, multi-step planning that remains challenging for monolithic policies. Such agents must jointly infer high-level plans, navigate through intermediate states, track progress over extended interaction histories, and execute visually grounded actions. Although prior work has explored modular multi-agent systems with a separate planner, the planner is typically not updated jointly with the actor. We investigate whether explicitly separating and jointly training planning and execution provides a stronger inductive bias for long-horizon web navigation. We introduce JHiMA-WebRL, a joint hierarchical multi-agent reinforcement learning framework for web agents that couples a task decomposer with an actor. To train the decomposer, we construct structured subtask supervision from multimodal GUI trajectories using a strong teacher model and use supervised fine-tuning to train both the decomposer and actor on trajectories augmented with generated subtasks. We then extend Group Relative Policy Optimization (GRPO) to jointly optimize the decomposer and actor in the interactive WebArena environment through hierarchical rollout grouping and role-specific advantages. On WebArena-Lite-v2, JHiMA-WebRL-14B, consisting of a 7B actor and a 7B decomposer, achieves a state-of-the-art success rate of 37.2%, exceeding the strongest reported baseline by 8.6 percentage points and surpassing larger 32B and 72B monolithic agents. These results suggest that jointly learned role factorization with smaller models can be a competitive alternative to scaling monolithic agents for long-horizon web tasks.

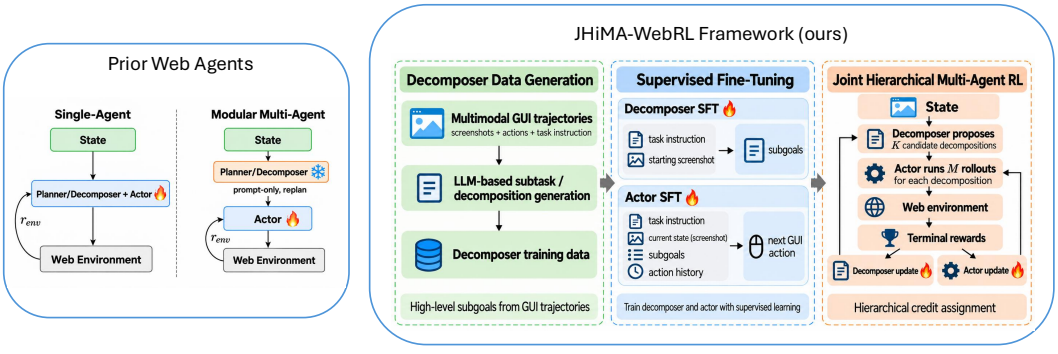



Figure 1: **Overview of JHiMA-WebRL.** Left: prior web agents use a single policy, or a separate planner that receives no RL updates while the actor is trained. Right: we generate subtask supervision from GUI trajectories, fine-tune the decomposer and actor, and jointly optimize both with hierarchical RL: $K$ decompositions, $M$ actor rollouts each, and role-specific advantages from terminal rewards.

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.

# 1 INTRODUCTION

Multimodal web agents interact with websites by perceiving screenshots, interpreting page elements, and executing actions such as clicking, typing, scrolling, and navigating between webpages. Such agents could support workflows in online shopping, web-based administration, repository management, and information retrieval. Benchmarks such as WebArena (Zhou et al., 2024) and VisualWebArena (Koh et al., 2024) evaluate this setting through tasks that require interactive execution across multiple steps. These tasks combine high-level planning with low-level visually grounded control: for example, to cancel an order, an agent must navigate intermediate pages, identify relevant interface elements, track progress over a long action history, and avoid repeated or irrelevant actions.

Many web agents use a monolithic policy that maps the task, observation, and action history directly to the next action, learning planning, state tracking, visual grounding, and execution within a single model. ScaleCUA (Liu et al., 2025a) and UI-TARS (Qin et al., 2025) demonstrate the value of large-scale GUI data for training such policies, while WebAgent-R1 (Wei et al., 2025) and WebRL (Qi et al., 2024) improve web interaction through online reinforcement learning (RL). Modular systems instead introduce a separate planner: Agent S uses a manager to generate subtasks from retrieved knowledge and experience for workers to execute (Agashe et al., 2025a), and Agent S2 adds proactive replanning as execution provides new observations (Agashe et al., 2025b). Even though an explicit planner module can support long-horizon behavior, the planner's parameters are not updated from the outcomes of action execution. Recent studies (Erdogan et al., 2025; Wu et al., 2026; Kong et al., 2026; Lu et al., 2025) focus on training modular agents for long-horizon tasks, but either train the modules separately, apply supervised learning or RL to only one module while freezing the other, or alternate updates between modules. To our knowledge, prior work has not studied simultaneous RL training of both planner and actor from a shared rollout batch in web environments.

We therefore introduce JHiMA-WebRL, a joint hierarchical multi-agent reinforcement learning framework for long-horizon web interaction. Figure 1 presents an overview of the three stages in our framework: decomposer data generation, supervised fine-tuning, and joint hierarchical multi-agent RL. First, a strong teacher model converts multimodal GUI trajectories into ordered subtasks, supplying planning annotations absent from existing datasets. Second, these annotations support supervised fine-tuning of two Qwen2.5-VL-7B models: a decomposer that predicts subtasks from the instruction and starting screenshot, and an actor that predicts the next GUI action conditioned on those subtasks. Finally, we jointly train both modules using terminal task success rewards while interacting with web environments, so that the decomposer learns plans suited to the actor's capabilities and the actor improves its execution of those plans. We extend Group Relative Policy Optimization (GRPO) through the hierarchical rollout structure: the decomposer proposes $K$ candidate decompositions, and the actor performs $M$ complete environment rollouts for each decomposition. Actor-level advantages compare executions under the same plan, while decomposer-level advantages compare plans by the average terminal reward of their executions, and both policies are updated from this shared batch without dense process rewards.

We evaluate JHiMA-WebRL-14B, our instantiation with a 7B actor and a 7B decomposer, on WebArena-Lite-v2 (Liu et al., 2025a), a refinement of the WebArena benchmark focusing on high-quality tasks. JHiMA-WebRL-14B achieves an average success rate of 37.2%, exceeding the best previously reported result by 8.6 percentage points and surpassing larger 32B and 72B models. Ablation studies further indicate that jointly updating the decomposer during RL is beneficial relative to updating the actor alone. These results suggest that joint training of role-factorized modules can be a competitive alternative to scaling monolithic agents.

# 2 RELATED WORK

WebArena, VisualWebArena, and WorkArena evaluate agents on web tasks requiring multi-step interaction, visual grounding, state tracking, and long-horizon planning (Zhou et al., 2024; Koh et al., 2024; Drouin et al., 2024). OSWorld, Windows Agent Arena, and MacArena study the broader setting of desktop and OS-level computer use (Xie et al., 2024; Bonatti et al., 2024; Muryn et al., 2026). Strong GUI agents such as UI-TARS and ScaleCUA train largely monolithic policies that

map observations and histories directly to actions using large-scale GUI data (Qin et al., 2025; Liu et al., 2025a). These systems demonstrate the importance of data scale and GUI grounding, but require a single policy to learn planning, progress tracking, grounding, and execution jointly, whereas we factorize web interaction into a trainable decomposer and actor.

A broad line of work studies modular or multi-agent LLM systems, including AutoGen, CAMEL, ChatDev, MetaGPT, and MAP (Wu et al., 2023; Li et al., 2023; Qian et al., 2024; Hong et al., 2024; Webb et al., 2025). In computer use, Agent S combines manager–worker decomposition with retrieved experience, and Agent S2 updates plans as subtasks are completed (Agashe et al., 2025a;b). OpAgent coordinates planning, grounding, reflection, and summarization, using execution feedback to guide subsequent instructions (Guo et al., 2026). These mechanisms concern the organization and adaptation of inference-time behavior. Related methods such as ReAct, Reflexion, MiRA, SEAgent, and TreeCUA explore intermediate reasoning, subgoals, feedback, or structured exploration (Yao et al., 2023; Shinn et al., 2023; Wang et al., 2026; Sun et al., 2025; Jiang et al., 2026). JHiMA-WebRL focuses on learning the decomposition interface from annotated trajectories and jointly adapting it with the actor through task rewards.

A complementary line of work studies training across planner–executor interfaces. Plan-and-Act separately fine-tunes planner and executor models using synthetic plans and action trajectories, and adds dynamic replanning at inference time (Erdogan et al., 2025). Our approach also uses trajectory-derived planning supervision, then jointly adapts both modules through RL. Navigating WebAI pairs a supervised hierarchical planner with an RL-trained controller (Thil et al., 2024). Planner Matters! optimizes only the planner using trajectory rewards from a VLM judge, freezing the actor and memory manager (Wu et al., 2026). Its joint-training ablation shares parameters across roles, whereas our decomposer and actor are separate policies. In robotic manipulation, HiRoC aligns an executor with subgoals through supervised fine-tuning, then applies hierarchical GRPO to the executor with a frozen planner, combining task-level and subgoal-level advantages (Kong et al., 2026). SWIRL instead trains both modules through alternating RL phases (Lu et al., 2025). In contrast, JHiMA-WebRL computes actor advantages within plans and decomposer advantages across plans, and updates both policies from a shared rollout batch.

PPO and GRPO provide the basic optimization tools for recent LLM and agent RL, while multi-agent RL methods such as MADDPG, COMA, QMIX, and MAPPO study credit assignment across interacting policies (Schulman et al., 2017; Shao et al., 2024; Lowe et al., 2017; Foerster et al., 2018; Rashid et al., 2018; Yu et al., 2022). For web and GUI agents, WebAgent-R1, WebRL, OpenWebRL, WorkForceAgent-R1, and OpAgent apply online multi-turn RL to browser-based tasks, usually optimizing a single agent policy from terminal or outcome-based rewards (Wei et al., 2025; Qi et al., 2024; Yang et al., 2026a; Zhuang et al., 2025; Guo et al., 2026). Other works improve credit assignment with process or progress signals, including Web-Shepherd, WebArbiter, TGPO, GraphGPO, and MiRA (Chae et al., 2025; Zhang et al., 2026; Chen et al., 2025; Cheng et al., 2026; Wang et al., 2026). We instead extend GRPO's rollout grouping and advantage computation to two jointly trained roles, using only terminal rewards and no dense process rewards.

# 3 METHOD

We propose JHiMA-WebRL, a joint hierarchical multi-agent reinforcement learning framework for long-horizon web navigation. Our approach factorizes interaction into two specialized but jointly optimized modules: a task decomposer and an actor. The decomposer maps the task instruction and initial screenshot into an ordered sequence of executable subtasks. The actor then conditions on the task, current observation, interaction history, and generated subtasks to produce visually grounded actions in the web interface. Below, we first describe the problem formulation, followed by the three stages of our framework: data generation for training the decomposer, supervised fine-tuning of both the decomposer and actor, and a joint hierarchical multi-agent extension of GRPO.

## 3.1 PROBLEM FORMULATION

We consider an interactive web environment modeled as a partially observed sequential decision problem. Each task is specified by a natural-language instruction $x$. At time step $t$, the agent receives a screenshot $o_t$ of the web interface and produces an action $a_t \in \mathcal{A}$, such as clicking a

page element, typing, scrolling, navigating between webpages, or terminating with an answer. The environment then transitions to a new state and returns the next observation $o_{t+1}$. A trajectory is denoted by $\tau = (o_0, a_0, o_1, a_1, \ldots, o_T)$, where $T$ is the trajectory length. Each trajectory receives a terminal task reward $R(\tau) \in \{0, 1\}$, where $R(\tau) = 1$ if the task is successfully completed and $R(\tau) = 0$ otherwise. Success is determined through environment-specific rule-based verification or an LLM judge.

Our modular policy consists of a decomposer policy $\pi_D \equiv \pi_{\theta_D}$ and an actor policy $\pi_A \equiv \pi_{\theta_A}$, parameterized by separate models. Given a task instruction $x$ and an initial observation $o_0$, the decomposer samples a decomposition

$$d = (g_1, g_2, \ldots, g_L) \sim \pi_D(\cdot \mid x, o_0), \tag{1}$$

where each $g_\ell$ is a natural-language subtask and $L$ is the number of generated subtasks. The actor then generates web-interface actions according to

$$a_t \sim \pi_A(\cdot \mid x, o_t, h_t, d), \tag{2}$$

where $h_t = (a_0, \ldots, a_{t-1})$ denotes the action history.

### 3.2 Decomposer Data Generation

Existing GUI trajectory datasets contain task instructions, screenshots, and action traces, but typically do not provide explicit subtask annotations. To train a task decomposer, we construct a new decomposition dataset from trajectory annotations in ScaleCUA (Liu et al., 2025a). ScaleCUA provides large-scale multimodal GUI data across platforms, including grounding annotations and human or agent trajectories.

For each trajectory, we provide a strong teacher model, GPT-5.2 Codex (OpenAI, 2025), with the original task instruction $x$ and starting screenshot $o_0$, together with the full sequence of user or agent actions $(a_0, \ldots, a_{T-1})$ and corresponding screenshots $(o_0, \ldots, o_T)$. A subgoal-generation guideline prompt, included in Appendix A, guides the annotation process.

The teacher produces an ordered list of executable subtasks $d = (g_1, \ldots, g_L)$ and optional descriptions of unnecessary or redundant actions in the trajectory. This process yields approximately 76K decomposer training examples; dataset statistics are provided in Table 3, and qualitative examples are shown in Appendix C. The resulting automated pipeline converts low-level multimodal GUI demonstrations into high-level planning supervision, enabling us to train an open-source decomposer rather than relying on a proprietary teacher at inference time.

### 3.3 Supervised Fine-Tuning

We first train both modules with supervised fine-tuning before reinforcement learning.

**Decomposer SFT.** The decomposer is initialized from the Qwen2.5-VL-7B model and trained on the generated subtask dataset. Given the task instruction and initial screenshot, the model is trained to predict the teacher-generated decomposition:

$$\mathcal{L}_D^{\mathrm{SFT}} = -\sum_{n=1}^{N_d} \log \pi_{\theta_D}\left(d^{(n)} \mid x^{(n)}, o_0^{(n)}\right), \tag{3}$$

where $N_d$ is the number of decomposer examples.

**Actor SFT.** The actor is also initialized from Qwen2.5-VL-7B and fine-tuned using supervised learning on released ScaleCUA trajectories augmented with the teacher-generated subtasks described in Section 3.2. For each trajectory, the actor receives the generated subtask list together with the task, current screenshot, and previous actions. It is then trained to predict the demonstrated next action:

$$\mathcal{L}_A^{\mathrm{SFT}} = -\sum_{n=1}^{N_A} \sum_{t=0}^{T_n-1} \log \pi_{\theta_A}\left(a_t^{(n)} \mid x^{(n)}, o_t^{(n)}, h_t^{(n)}, d^{(n)}\right), \tag{4}$$

where $N_A$ is the number of actor trajectories. This trains the actor to use decompositions as an explicit planning context while learning GUI grounding and action generation from the released ScaleCUA trajectories.

### 3.4 JOINT HIERARCHICAL MULTI-AGENT GRPO

After supervised fine-tuning, the joint RL stage of JHiMA-WebRL optimizes the decomposer and actor using GRPO (Shao et al., 2024), extended to their hierarchical interaction. We retain the GRPO clipped policy objectives and compute role-specific advantages from hierarchical rollout groups to assign credit to planning and execution. Standard web-agent RL methods often optimize a single policy from terminal trajectory rewards (Wei et al., 2025; Qi et al., 2024; Yang et al., 2026a). In contrast, our setting contains two interacting policies with different roles and action spaces: the decomposer generates a high-level plan, and the actor executes low-level web-interface actions conditioned on that plan.

For a task $x$ with initial observation $o_0$, the decomposer first samples $K$ candidate decompositions $d_i \sim \pi_{\theta_D}(\cdot \mid x, o_0)$, $i = 1, \ldots, K$. For each $d_i$, the actor then generates $M$ independent multi-turn trajectories $\tau_{i,j} \sim \pi_{\theta_A}(\cdot \mid x, o_0, d_i)$, $j = 1, \ldots, M$, by interacting with the environment. Following Section 3.1, a trajectory is a sequence of GUI actions, $\tau_{i,j} = \left(a_{i,j,0}, \ldots, a_{i,j,T_{i,j}-1}\right)$ of length $T_{i,j}$, where $a_{i,j,k}$ denotes the action generated at environment interaction step $k$. Because each action is produced autoregressively by the VLM, we further write $a_{i,j,k} = \left(a_{i,j,k,1}, \ldots, a_{i,j,k,|a_{i,j,k}|}\right)$, where $u$ indexes the generated tokens $a_{i,j,k,u}$ within the action. Each actor trajectory receives a terminal binary reward $R_{i,j} = R(\tau_{i,j}) \in \{0, 1\}$.

**Actor-level advantage.** For a fixed decomposition $d_i$, we compute the group-relative actor advantage by comparing the $M$ downstream trajectories generated under the same decomposition:

$$A^A_{i,j} = \frac{R_{i,j} - \frac{1}{M}\sum_{m=1}^{M} R_{i,m}}{\mathrm{std}_m\left[R_{i,m}\right] + \delta}, \tag{5}$$

where $\delta > 0$ is a small constant for numerical stability. Thus, all action tokens in trajectory $\tau_{i,j}$ share the same trajectory-level advantage $A^A_{i,j}$.

Following the token-level M-GRPO objective (Wei et al., 2025), we define the actor importance-sampling ratio for token $u$ of action $a_{i,j,k}$ as

$$r^A_{i,j,k,u}(\theta_A) = \frac{\pi_{\theta_A}\left(a_{i,j,k,u} \mid c^A_{i,j,k}, a_{i,j,k,<u}\right)}{\pi_{\theta^{\mathrm{old}}_A}\left(a_{i,j,k,u} \mid c^A_{i,j,k}, a_{i,j,k,<u}\right)}, \tag{6}$$

where $c^A_{i,j,k} = (x, o_{i,j,k}, h_{i,j,k}, d_i)$ contains the task instruction, current observation, action history available at environment step $k$, and decomposition $d_i$.

The clipped token-level actor objective is

$$\begin{aligned}\mathcal{L}_A = -\frac{1}{KM}\sum_{i=1}^{K}\sum_{j=1}^{M}\frac{1}{T_{i,j}}\sum_{k=0}^{T_{i,j}-1}\frac{1}{|a_{i,j,k}|}\sum_{u=1}^{|a_{i,j,k}|}\Big[&\min\left(r^A_{i,j,k,u}(\theta_A)A^A_{i,j},\right.\\ &\left.\mathrm{clip}(r^A_{i,j,k,u}(\theta_A), 1-\epsilon_A, 1+\epsilon_A)A^A_{i,j}\right)\\ &-\beta_A D^A_{\mathrm{KL}}(i,j,k,u)\Big],\end{aligned} \tag{7}$$

where $\epsilon_A$ is the clipping parameter and $D^A_{\mathrm{KL}}(i,j,k,u)$ is the per-token KL divergence between $\pi_{\theta_A}$ and the reference (SFT) actor, weighted by $\beta_A$.

**Decomposer-level advantage.** To evaluate decomposition quality, we first aggregate the downstream rewards of the $M$ actor trajectories conditioned on $d_i$:

$$S_i = \frac{1}{M}\sum_{j=1}^{M} R_{i,j}. \tag{8}$$

The decomposer advantage is then computed relative to the $K$ candidate decompositions sampled for the same task:

$$A^D_i = \frac{S_i - \frac{1}{K}\sum_{\ell=1}^{K} S_\ell}{\mathrm{std}_\ell[S_\ell] + \delta}. \tag{9}$$

Because $d_i$ is itself an autoregressively generated sequence, we define the token-level ratio $r^D_{i,u}(\theta_D) = \pi_{\theta_D}(d_{i,u} \mid x, o_0, d_{i,<u}) / \pi_{\theta^{\text{old}}_D}(d_{i,u} \mid x, o_0, d_{i,<u})$ for its $u$-th token, analogously to Eq. equation 6. The token-level decomposer objective is

$$\mathcal{L}_D = -\frac{1}{K}\sum_{i=1}^{K}\frac{1}{|d_i|}\sum_{u=1}^{|d_i|}\Big[\min\big(r^D_{i,u}(\theta_D)A^D_i, \\ \text{clip}(r^D_{i,u}(\theta_D), 1-\epsilon_D, 1+\epsilon_D)A^D_i\big) \\ -\beta_D D^D_{\text{KL}}(i,u)\Big], \tag{10}$$

with $\epsilon_D$, $\beta_D$, and $D^D_{\text{KL}}$ defined analogously to those for the actor.

Thus, the downstream execution quality of a decomposition determines a single group-relative advantage $A^D_i$, which is propagated to all generated tokens of that decomposition. This enables the decomposer to learn which planning outputs are most useful to the actor while retaining token-level policy optimization.

Finally, the two modules are optimized jointly according to

$$\mathcal{L}_{\text{JHiMA}} = \mathcal{L}_A + \mathcal{L}_D. \tag{11}$$

Since $\theta_A$ and $\theta_D$ are disjoint, this sum amounts to updating each module with its own objective, using the same rollout batch.

Appendix B shows that $A^D_i$ corresponds to a group-baselined policy-gradient estimate for the decomposer under the shared task reward.

# 4 Experiments

## 4.1 Datasets

The task decomposer was trained with supervised fine-tuning using approximately 76K task-decomposition examples, which were generated from ScaleCUA trajectories using GPT-5.2 Codex as described in Section 3.2. The actor was fine-tuned on all ScaleCUA trajectories augmented with the teacher-annotated subtasks, along with grounding data spanning web, mobile, and desktop platforms, resulting in 2,535,552 training examples. Unlike standard ScaleCUA SFT, the actor thus receives a task decomposition in its context as a planning scaffold.

Both modules were then jointly trained through online reinforcement learning using the training tasks associated with WebArena-Lite, following the setup of WebAgent-R1 (Wei et al., 2025). WebArena contains 812 interactive web tasks spanning five domains—Reddit, GitLab, e-commerce administration, maps, and online shopping. WebArena-Lite selects 165 human-verified tasks for evaluation, leaving 647 tasks from the original WebArena benchmark for reinforcement-learning interaction. No WebArena-Lite evaluation task (and hence no WebArena-Lite-v2 task) is used during SFT or RL. We also employed the flexible evaluation protocol of Liu et al. (2025a), including LLM-based fuzzy matching using GPT-5.2 Codex, to compute the binary task outcome reward at the end of web interaction.

For evaluation, we used WebArena-Lite-v2, introduced by ScaleCUA (Liu et al., 2025a). WebArena-Lite-v2 is a refinement of the evaluation subset of WebArena-Lite designed specifically for evaluating purely visual GUI agents. It contains 154 tasks across five website environments: Shopping, ShoppingAdmin, Reddit, GitLab, and OpenStreetMap. The benchmark includes heterogeneous task objectives such as question answering, navigation, page-content matching, and state-changing operations, with success determined by the flexible evaluation criteria of Liu et al. (2025a).

## 4.2 Baselines

We compared against native agents at several model scales ranging from 7B to 72B, allowing comparisons between our 14B modular system and both smaller and larger monolithic systems. We also compared against multi-agent systems consisting of a proprietary planner (GPT-4o) with a grounding module. These provide comparisons with systems that separate planning from visual grounding,

but do not jointly train a learned decomposer and actor as in JHiMA-WebRL-14B. All baseline values are taken from ScaleCUA (Liu et al., 2025a), except for the reproduced ScaleCUA results and the GUI-Libra-8B model reported by Yang et al. (2026b). We also trained a ScaleCUA-32B RL baseline, specifically fine-tuning the SFT model on the WebArena training set, using standard multi-turn GRPO with 7 rollouts and a maximum of 15 interaction steps. For a fair comparison with our method, we only included baselines that do not use HTML or DOM information.

### 4.3 Experimental Details

We trained the 7B decomposer and 7B actor of JHiMA-WebRL-14B in two stages: SFT, followed by joint hierarchical RL. The decomposer was initialized from Qwen2.5-VL-7B and fine-tuned on the approximately 76K decomposition examples generated from ScaleCUA multimodal trajectories, on 8 NVIDIA H100 GPUs with 80 GB of memory each via full-parameter fine-tuning. We trained for `1` epoch with a learning rate of `1e-5` and a `cosine` learning-rate schedule with a warmup ratio of `0.03`. The per-device batch size was `2`, with gradient accumulation over `4` steps, resulting in an effective global batch size of `64`. We used DeepSpeed (Rajbhandari et al., 2020) for distributed training with ZeRO-3, along with gradient checkpointing, bf16 mixed-precision training, gradient clipping to a maximum norm of `1`, and a maximum token length of `30720`. The actor was also initialized from Qwen2.5-VL-7B and fine-tuned with the same hyperparameters as above, using released ScaleCUA trajectories augmented with generated subtasks.

After SFT, both the actor and decomposer were jointly optimized through LoRA fine-tuning using GRPO, with standard-deviation and response-length normalization removed to reduce optimization bias, following Liu et al. (2025b). We used LoRA rank $r = 32$ and scaling factor $\alpha = 64$ for both modules, with a separate adapter for each. For each task, the decomposer sampled $K = 3$ candidate decompositions. For every decomposition, the actor generated $M = 7$ complete trajectories. For both, we used `0.8` as the sampling temperature, and we allowed a maximum of `15` interaction steps per trajectory. We optimized the LoRA parameters with a learning rate of `1e-5`. The GRPO updates used clipping parameters $\epsilon_A = \epsilon_D = 0.2$, KL coefficients $\beta_A = \beta_D = 0$ (i.e., no KL penalty), and `2` optimization steps per rollout batch. We used `32` gradient accumulation steps and trained for `1` epoch over the 647 training tasks. We used the efficient parallel trajectory rollout generation of M-GRPO (Wei et al., 2025) for each task decomposition, with training taking approximately 720 GPU-hours on a cluster of 8 H100 GPUs with 80 GB of memory each.

**Evaluation.** We report the task success rate on all 154 WebArena-Lite-v2 tasks, averaged over 4 independent evaluation runs. We sampled one decomposition at test time using a temperature of 0.0 and allowed a maximum of 15 interaction steps, consistent with the baselines.

Table 1: Success rates (%) on WebArena-Lite-v2. Our results are averaged over 4 runs; baseline results are taken from Liu et al. (2025a) and Yang et al. (2026b), except for ScaleCUA-32B RL trained by us. Dashes indicate domain scores not reported.

| Method | GitLab | Map | Reddit | Shopping | ShoppingAdmin | Average |
|---|---|---|---|---|---|---|
| UI-TARS-1.5-7B | – | – | – | – | – | 20.8 |
| UI-TARS-72B-DPO | – | – | – | – | – | 23.4 |
| ScaleCUA-7B SFT | 28.3 | 15.4 | 27.6 | 18.8 | 30.7 | 23.9 |
| ScaleCUA-32B SFT | 34.2 | 10.6 | 26.3 | 16.5 | 33.6 | 24.0 |
| ScaleCUA-32B RL | 35.0 | 10.6 | 21.0 | 24.4 | 25.7 | 24.0 |
| GUI-Libra-8B | 31.7 | **17.3** | 35.5 | 26.1 | 25.0 | 26.6 |
| GPT-4o Planner + UI-TARS-1.5-7B Grounder | – | – | – | – | – | 28.6 |
| GPT-4o Planner + ScaleCUA-7B Grounder | – | – | – | – | – | 28.6 |
| **JHiMA-WebRL-14B (Ours)** | **47.5** | 12.5 | **60.5** | **29.0** | **44.3** | **37.2** |

### 4.4 Results

As shown in Table 1, JHiMA-WebRL-14B achieved the highest average success rate of 37.2% over four runs aggregated across the five websites, also achieving the highest scores on GitLab (47.5%), Reddit (60.5%), Shopping (29.0%), and ShoppingAdmin (44.3%). This exceeded the best previously reported result on WebArena-Lite-v2, the GPT-4o planner–grounder baselines (28.6%), by 8.6 percentage points. Relative to ScaleCUA-7B SFT (23.9%), ScaleCUA-32B SFT/RL (24.0%), and

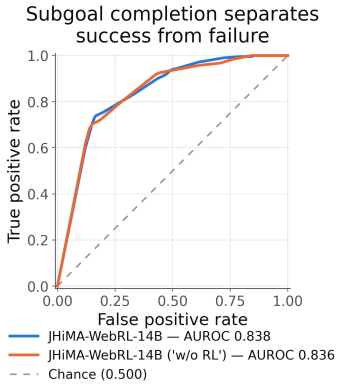
UI-TARS-72B-DPO (23.4%), the gains were 13.3, 13.2, and 13.8 percentage points, respectively. Thus, the 14B modular system outperformed the larger ScaleCUA-32B SFT/RL and UI-TARS-72B-DPO systems. The Map domain is the exception, where JHiMA-WebRL-14B (12.5%) trails GUI-Libra-8B (17.3%) and ScaleCUA-7B SFT (15.4%), primarily because the agent repeats irrelevant actions and becomes stuck in loops (Figure 6). These results support learned role factorization with smaller, jointly trained models as a competitive alternative to scaling a monolithic agent.

Table 2: Ablation success rates (%) on WebArena-Lite-v2, averaged over 4 runs. "Rollouts" denotes the number of complete trajectories per training task.

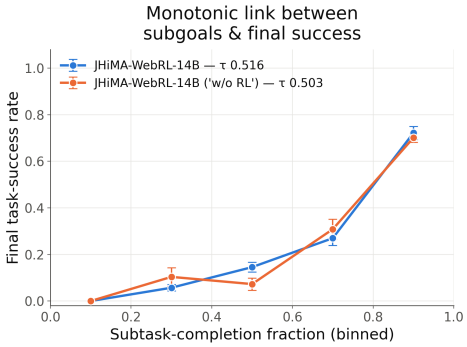
| Method | GitLab | Map | Reddit | Shopping | ShoppingAdmin | Average |
|---|---|---|---|---|---|---|
| **JHiMA-WebRL-14B (Ours)** | **47.5** | 12.5 | **60.5** | 29.0 | 44.3 | **37.2** |
| w/o decomposer RL | 37.5 | 10.6 | 54.0 | 25.6 | 45.7 | 33.4 |
| w/o decomposer (actor-only RL, 21 rollouts) | 44.2 | 14.4 | 51.3 | 26.7 | 43.6 | 34.9 |
| w/o decomposer (actor-only RL, 7 rollouts) | 39.2 | **15.4** | 48.7 | **30.1** | 45.7 | 35.2 |
| w/o RL (SFT decomposer + SFT actor) | 43.3 | 8.6 | 59.2 | 25.6 | **46.4** | 35.1 |

### 4.5 Ablation Study

We analyzed the contribution of the components of JHiMA-WebRL-14B through ablations (Table 2). First, we trained only the actor with RL while keeping the SFT decomposer frozen ('w/o decomposer RL'), using the same number of rollouts per task ($K$=3, $M$=7) as JHiMA-WebRL-14B. This reduced the average success rate to 33.4%, a drop of 3.8 percentage points, indicating that updating the decomposer jointly with the actor matters. Second, to assess the role of the decomposer, we trained the actor alone using standard multi-turn GRPO without subtask conditioning. The actor was initialized from Qwen2.5-VL-7B and fine-tuned using supervised learning on released ScaleCUA trajectories and grounding data, without any subtasks. We trained two variants: one with the same total number of rollouts per task as JHiMA-WebRL-14B (21), and one with 7 rollouts, matching the per-decomposition group size of our actor. The two variants reached 34.9% and 35.2%, i.e., 2.3 and 2.0 percentage points below JHiMA-WebRL-14B. Finally, we removed RL entirely and evaluated the SFT decomposer and SFT actor together ('w/o RL'), which reduced the average success rate by 2.1 percentage points.




(a) ROC curve using the fraction of completed subgoals as a score for predicting final task success.




(b) Empirical task success rate as a function of subgoal completion.

Figure 2: **Subgoal quality analysis.** (a) Discriminative power of subgoal completion for predicting terminal task success. (b) Monotonic relationship between progress through the generated decomposition and empirical task success.

### 4.6 Subgoal Quality Analysis

We also analyzed whether the generated decompositions provide meaningful intermediate structure. Following the subgoal-quality analysis of MiRA (Wang et al., 2026), we measured whether partial

subgoal completion was predictive of eventual task success and whether task success increased monotonically as more subgoals were completed.

For each WebArena-Lite-v2 trajectory from the four evaluation runs, we used GPT-5.2 Codex to judge whether each subgoal generated by the decomposer was completed, based on the sequence of predicted actions and resulting screenshots, and computed the fraction of generated subgoals that were completed. We first evaluated whether the fraction of completed subgoals discriminated successful trajectories from failed ones. We treated the fraction as a continuous prediction score for the binary task outcome and computed the area under the receiver operating characteristic curve (AUROC). As shown in Figure 2a, the subgoals generated by JHiMA-WebRL-14B achieved an AUROC of 0.838, well above chance (0.5), and slightly higher than without RL (0.836). This indicates that progress through the generated decompositions is strongly aligned with downstream task success.

A useful decomposition should also provide a calibrated notion of progress, so we grouped trajectories by subgoal completion fraction and computed the empirical success rate per group. Figure 2b shows a clear increase in empirical task success with subgoal completion fraction, with Kendall's rank correlation coefficient $\tau = 0.516$ for JHiMA-WebRL-14B, improving slightly after joint RL. Together with the AUROC analysis, these results suggest that the generated decompositions are not merely plausible natural-language plans, but encode intermediate milestones that are increasingly associated with downstream task completion.

## 5 Limitations

Our study has several limitations. (i) *Train–test overlap:* RL is performed on the remaining WebArena tasks, which are disjoint from the evaluation tasks but can share intent templates and websites with them; results thus reflect in-environment generalization, not transfer to unseen websites. (ii) *Supervision and cost:* decomposition supervision is generated by a proprietary teacher (GPT-5.2 Codex) without human verification of annotation quality, and hierarchical rollouts require $K \times M$ complete episodes per task, which is costly in live environments. (iii) *Open-loop planning:* the decomposer plans once from the initial observation and cannot revise the plan during execution. (iv) *Coarse credit:* terminal rewards assign the same advantage to every token of a decomposition or trajectory, without identifying which subtask or action caused success or failure.

## 6 Conclusion and Future Directions

We introduced JHiMA-WebRL, a framework for jointly training a task decomposer and a visually grounded actor. The framework generates subtask supervision from multimodal GUI trajectories to train the decomposer and extends GRPO with hierarchical rollout grouping and role-specific advantages. JHiMA-WebRL-14B achieved an average success rate of 37.2% on WebArena-Lite-v2, surpassing the best previously reported result by 8.6 percentage points and outperforming the larger ScaleCUA-32B SFT/RL and UI-TARS-72B-DPO baselines. Ablations indicate that subtask-augmented SFT, the decomposer, and joint updates to the decomposer during RL each contribute to the final performance.

Our work presents several future directions. Firstly, the decomposer could be made *adaptive*, revising subtasks as execution deviates from the plan. Secondly, joint optimization can be combined with intermediate progress signals such as milestone rewards (Wang et al., 2026) or graph-based credit assignment (Cheng et al., 2026). Finally, jointly trained decomposer–actor systems could be extended to cross-domain and live-web settings, where online RL has recently been shown to be feasible (Yang et al., 2026a). This would test whether modular role factorization transfers better than monolithic policies.

### AI use statement

Generative AI is part of our method: GPT-5.2 Codex is used as a teacher model to generate subtask annotations from ScaleCUA trajectories (Section 3.2) and as a judge of subgoal completion in the analysis of Section 4.6; the evaluation protocol of Liu et al. (2025a) also uses an LLM for fuzzy answer matching. We used Claude Opus 4.8 for coding and GPT-5.6/6 to assist with drafting the

paper and reviewing the literature. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### Reproducibility statement

Section 3 specifies the full training objective, including the hierarchical advantages (Eqs. 5–9) and the clipped objectives for both modules. Section 4.1 describes the training and evaluation data, and Section 4.3 lists all SFT and RL hyperparameters. The exact prompt used to generate decomposition supervision is given in Appendix A. We will also release the code, the decomposition dataset, and model checkpoints.

## References

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: An open agentic framework that uses computers like a human. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2410.08164.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents, 2025b. URL https://arxiv.org/abs/2504.00906.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows agent arena: Evaluating multi-modal os agents at scale. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2409.08264.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beongwoo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-shepherd: Advancing prms for reinforcing web agents, 2025. URL https://arxiv.org/abs/2505.15277.

Ziyuan Chen, Zhenghui Zhao, Zhangye Han, Miancan Liu, Xianhang Ye, Yiqing Li, Hongbo Min, Jinkui Ren, Xiantao Zhang, and Guitao Cao. TGPO: Tree-guided preference optimization for robust web agent reinforcement learning, 2025. URL https://arxiv.org/abs/2509.14172.

Xin Cheng, Shuo He, Lang Feng, HaiYang Xu, Ming Yan, Lei Feng, and Bo An. Beyond trajectory-level attribution: Graph-based credit assignment for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2605.26684.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-Act: Improving planning of agents for long-horizon tasks, 2025. URL https://arxiv.org/abs/2503.09572.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1705.08926.

Yuyu Guo, Wenjie Yang, Siyuan Yang, Ziyang Liu, Cheng Chen, Yuan Wei, Yun Hu, Yang Huang, Guoliang Hao, Dongsheng Yuan, Jianming Wang, Xin Chen, Hang Yu, Lei Lei, and Peng Di. Opagent: Operator agent for web navigation, 2026. URL https://arxiv.org/abs/2602.13559.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.00352.

Deyang Jiang, Jing Huang, Xuanle Zhao, Lei Chen, Liming Zheng, Fanfan Liu, Haibo Qiu, Peng Shi, and Zhixiong Zeng. Treecua: Efficiently scaling gui automation with tree-structured verifiable evolution, 2026. URL https://arxiv.org/abs/2602.09662.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024. URL https://arxiv.org/abs/2401.13649.

He Kong, Zengjue Chen, Qi Wang, Qianli Xing, Runliang Niu, Peidong Liu, Jiawei Li, Shiqi Wang, and Yi Chang. Beyond flat policies: Hierarchical post-training for embodied agents in robotic manipulation, 2026. URL https://arxiv.org/abs/2608.05999.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. *arXiv preprint arXiv:2303.17760*, 2023. URL https://arxiv.org/abs/2303.17760.

Zhaoyang Liu, Jingjing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, Shenglong Ye, Qingyun Li, Xuan Dong, Yue Yu, Chenyu Lu, YunXiang Mo, Yao Yan, Zeyue Tian, Xiao Zhang, Yuan Huang, Yiqian Liu, Weijie Su, Gen Luo, Xiangyu Yue, Biqing Qi, Kai Chen, Bowen Zhou, Yu Qiao, Qifeng Chen, and Wenhai Wang. Scalecua: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*, 2025a. URL https://github.com/OpenGVLab/ScaleCUA.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025b. URL https://arxiv.org/abs/2503.20783.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1706.02275.

Quanfeng Lu, Zhantao Ma, Shuai Zhong, Jin Wang, Dahai Yu, Michael K. Ng, and Ping Luo. SWIRL: A staged workflow for interleaved reinforcement learning in mobile gui control, 2025. URL https://arxiv.org/abs/2508.20018.

Victor Muryn, Maksym Shamrai, Sofiia Mazepa, and Yehor Khodysko. Macarena: Benchmarking computer use agents on an online macos environment, 2026. URL https://arxiv.org/abs/2606.06560.

OpenAI. Introducing gpt-5.2-codex. https://openai.com/index/introducing-gpt-5-2-codex/, December 2025. Accessed: 2026-09-25.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Xinyue Yang, Jiadai Sun, Yu Yang, Shuntian Yao, Tianjie Zhang, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024. URL https://arxiv.org/abs/2411.02337.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2307.07924.

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. URL https://arxiv.org/abs/2501.12326.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020. doi: 10.1109/SC41405.2020.00024.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4295–4304. PMLR, 2018. URL https://proceedings.mlr.press/v80/rashid18a.html.

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

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. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2303.11366.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. SEAgent: Self-evolving computer use agent with autonomous learning from experience, 2025. URL https://arxiv.org/abs/2508.04700.

Lucas-Andreï Thil, Mirela Popa, and Gerasimos Spanakis. Navigating WebAI: Training agents to complete web tasks with large language models and reinforcement learning. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024. doi: 10.1145/3605098.3635903. URL https://arxiv.org/abs/2405.00516.

Taiyi Wang, Sian Gooding, Florian Hartmann, Oriana Riva, and Edward Grefenstette. A subgoal-driven framework for improving long-horizon llm agents, 2026. URL https://arxiv.org/abs/2603.19685.

Taylor Webb, Shanka Subhra Mondal, and Ida Momennejad. A brain-inspired agentic architecture to improve planning with LLMs. *Nature Communications*, 16:8633, 2025. doi: 10.1038/s41467-025-63804-5. URL https://www.nature.com/articles/s41467-025-63804-5.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. In *The 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://arxiv.org/abs/2505.16421.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023. URL https://arxiv.org/abs/2308.08155.

Wenyi Wu, Sibo Zhu, Kun Zhou, and Biwei Huang. Planner Matters! an efficient and unbalanced multi-agent collaboration framework for long-horizon planning, 2026. URL https://arxiv.org/abs/2605.02168.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

Rui Yang, Qianhui Wu, Yuxi Chen, Hao Bai, Wenlin Yao, Hao Cheng, Baolin Peng, Huan Zhang, Tong Zhang, and Jianfeng Gao. Openwebrl: Demystifying online multi-turn reinforcement learning for visual web agents, 2026a. URL https://arxiv.org/abs/2606.02031.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baoling Peng, Huan Zhang, Jianfeng Gao, and Tong Zhang. Gui-libra: Training native gui agents to reason and act with action-aware supervision and partially verifiable rl, 2026b. URL https://arxiv.org/abs/2602.22190.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2103.01955.

Yao Zhang, Shijie Tang, Zeyu Li, Zhen Han, and Volker Tresp. Webarbiter: A principle-guided reasoning process reward model for web agents, 2026. URL https://arxiv.org/abs/2601.21872.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://webarena.dev.

Yuchen Zhuang, Di Jin, Jiaao Chen, Wenqi Shi, Hanrui Wang, and Chao Zhang. Workforceagent-r1: Incentivizing reasoning capability in llm-based web agents via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.22942.

# APPENDIX

Table 3: Subtask-decomposition dataset statistics.

| Platform | # Examples | Avg. # Subtasks | Frac. w/ Unnecessary Actions |
|---|---|---|---|
| Windows | 5,873 | 2.68 | 1.2% |
| macOS | 3,845 | 2.43 | 1.1% |
| Linux | 25,122 | 2.20 | 1.8% |
| Android | 17,245 | 2.84 | 1.6% |
| Web | 24,311 | 2.73 | 1.5% |
| Total | 76,396 | 2.56 | 1.6% |

## A SUBTASK GENERATION PROMPT

The following prompt is provided together with the task instruction and interleaved trajectory screenshots. Images are inserted as multimodal image content immediately after the corresponding textual markers.

```
You are an AI assistant tasked with generating high level subtasks that highlight key subgoals
      necessary to complete a task.

Task: {TASK}

Starting Image URL:
{STARTING_IMAGE}

Below are the actions taken by the user at each step to perform the task

Step 1: {STEP_1}
Image URL after step 1:
{IMAGE_AFTER_STEP_1}

Step 2: {STEP_2}
Image URL after step 2:
{IMAGE_AFTER_STEP_2}

...

Step N: {STEP_N}
Image URL after step N:
{IMAGE_AFTER_STEP_N}

Guidelines for Subtask Generation
    1. Identify Essential High-Level Subgoals:
    - A subgoal should represent a significant step involving user action that leads to
    noticeable page transitions or meaningful changes in system state (i.e Image).
    - Consolidate closely related user actions (such as applying multiple filters or selecting
      several options) into a single subgoal, rather than separate subtasks for each action.
    - Prioritize only the most critical actions necessary for meaningful progression,
    avoiding the inclusion of minor or unnecessary steps (e.g., scroll, hover).

    2. Provide a Concise Subgoal Analysis:
    - Before creating the subtasks, offer a brief paragraph summarizing the main subgoals,
    emphasizing significant transitions or page-level actions.

    3. Ensure Clear Goal:
    - If multiple related actions occur (e.g., setting filters 1, 2, and 3), combine them into
      one
    subgoal with clear criteria verifying all required conditions.
    - The subtasks should contain only essential steps, explicitly excluding unnecessary
    actions, and should not exceed five critical subtasks. It is not necessary to use all five
    subtasks if fewer steps adequately represent the essential subgoals.

    Finally, based on the generated subtasks and subgoal analysis, report if there are any
     unnecessary or redundant actions which aren't required to perform the task.
    If all the actions are important say <NONE>, else say <YES> alongwith the unnecessary
     actions.

    ### Output Format
    Before generating the subtasks, first produce a concise subgoal analysis in a single
```

```
paragraph summarizing the required actions. Then, based on this, generate the subtasks
following the format below:

[SUBGOAL ANALYSIS]
[One-paragraph summary explaining the key subgoals and their logical
sequence in task completion.]

[SUBTASKS]
SUBTASK X:
[Short title of the action/goal]
- Goal:
[Brief description of the subgoal at this stage, emphasizing the
purpose of the action.]

[UNNECESSARY ACTIONS]
[NONE or YES alongwith the unnecessary actions]
```

## B Policy-Gradient Interpretation of the Hierarchical Advantages

Both modules share the single objective $J(\theta_D, \theta_A) = \mathbb{E}_{d\sim\pi_{\theta_D}} \mathbb{E}_{\tau\sim\pi_{\theta_A}(\cdot|x,o_0,d)}[R(\tau)]$, so the setting is fully cooperative with a shared reward. By the policy-gradient theorem, $\nabla_{\theta_D} J = \mathbb{E}_d\big[\nabla_{\theta_D} \log \pi_{\theta_D}(d \mid x, o_0)\,(Q(d) - b)\big]$ with $Q(d) = \mathbb{E}_\tau[R(\tau) \mid d]$ and any baseline $b$ that does not depend on $d$. The score $S_i$ in Eq. equation 8 is an unbiased Monte Carlo estimate of $Q(d_i)$ under the current actor, and subtracting the group mean yields a GRPO-style group baseline; the $M$ inner rollouts reduce the variance of this estimate relative to scoring a decomposition by a single rollout. Likewise, conditioning the actor baseline on $d_i$ (Eq. equation 5) removes the reward variance attributable to plan quality from the actor's advantage, so that actor updates compare executions of the same plan. Because both policies change simultaneously, each module's learning problem is non-stationary; we rely on the PPO-style clipping and a small number of optimization steps per batch to keep per-iteration policy changes small.

## C Qualitative Examples

Figures 3 and 4 show teacher-generated annotations for two ScaleCUA trajectories, one containing an unnecessary action and one in which all actions are necessary. Figure 5 shows a decomposition produced by our decomposer from the task instruction and starting screenshot of a WebArena ShoppingAdmin task. Figure 6 shows a failure case for a Map task.

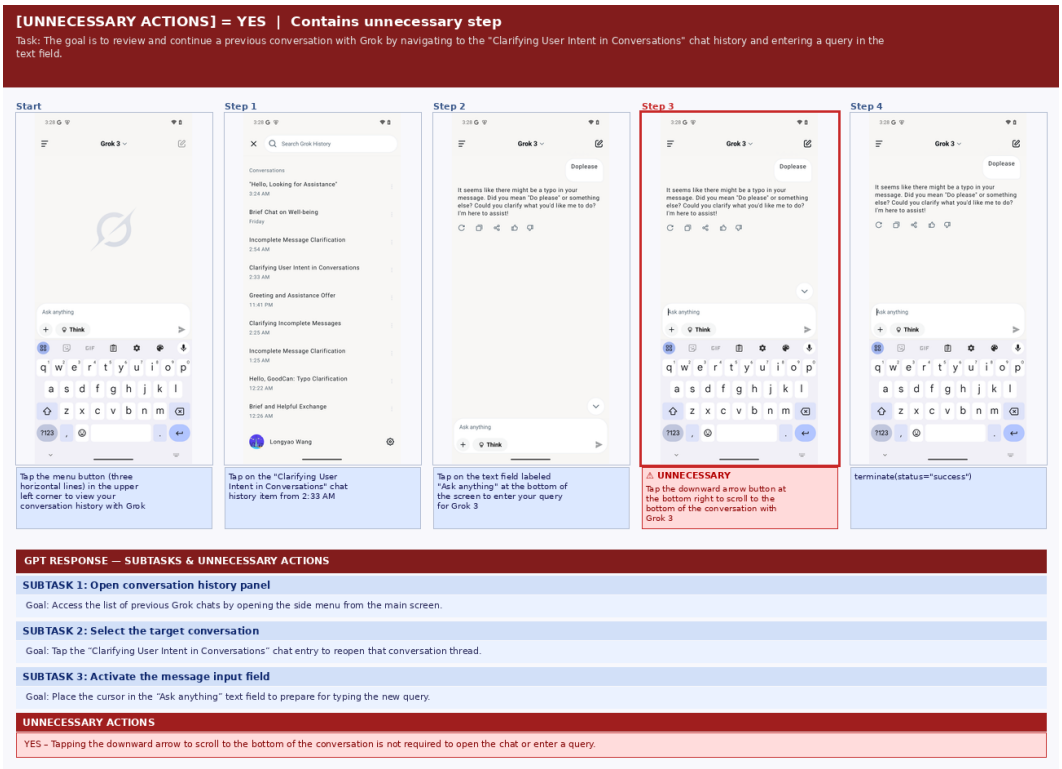



Figure 3: Teacher annotation of a ScaleCUA mobile trajectory with an unnecessary action (Step 3, highlighted). The teacher outputs three subtasks and flags the redundant scroll action.

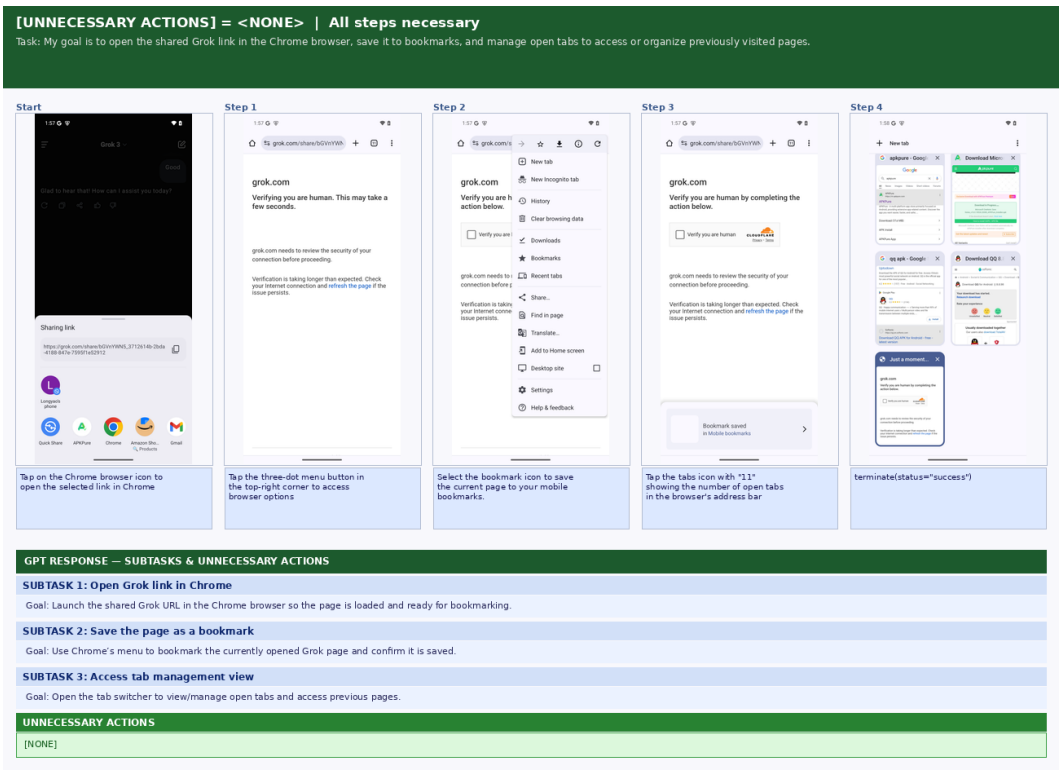
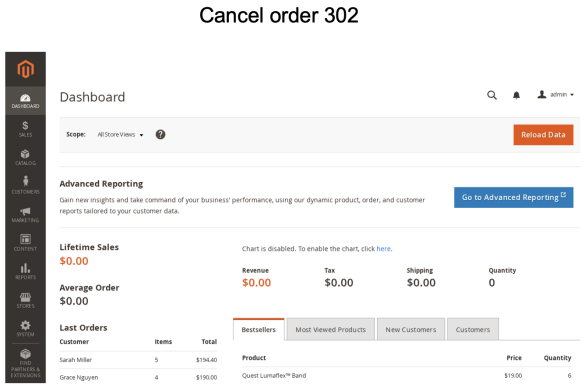



Figure 4: Teacher annotation of a ScaleCUA mobile trajectory in which all actions are necessary; the teacher outputs three subtasks and `[NONE]` for unnecessary actions.




```
SUBTASK 1: Open the Orders list
 – Goal: Navigate from the dashboard to the Sales > Orders page where all orders can be managed.
SUBTASK 2: Locate order 302
– Goal: Use the filter panel to search by order ID “302” and apply the filter so the correct order appears.
SUBTASK 3: Open order details
– Goal: Click “View” for order #302 to access its detailed order page.
SUBTASK 4: Cancel and confirm the order
– Goal: Click “Cancel” on the order page and confirm the cancellation in the dialog."
```

Figure 5: Decomposition generated by the JHiMA-WebRL decomposer for the ShoppingAdmin evaluation task “Cancel order 302” from the task instruction and starting screenshot.

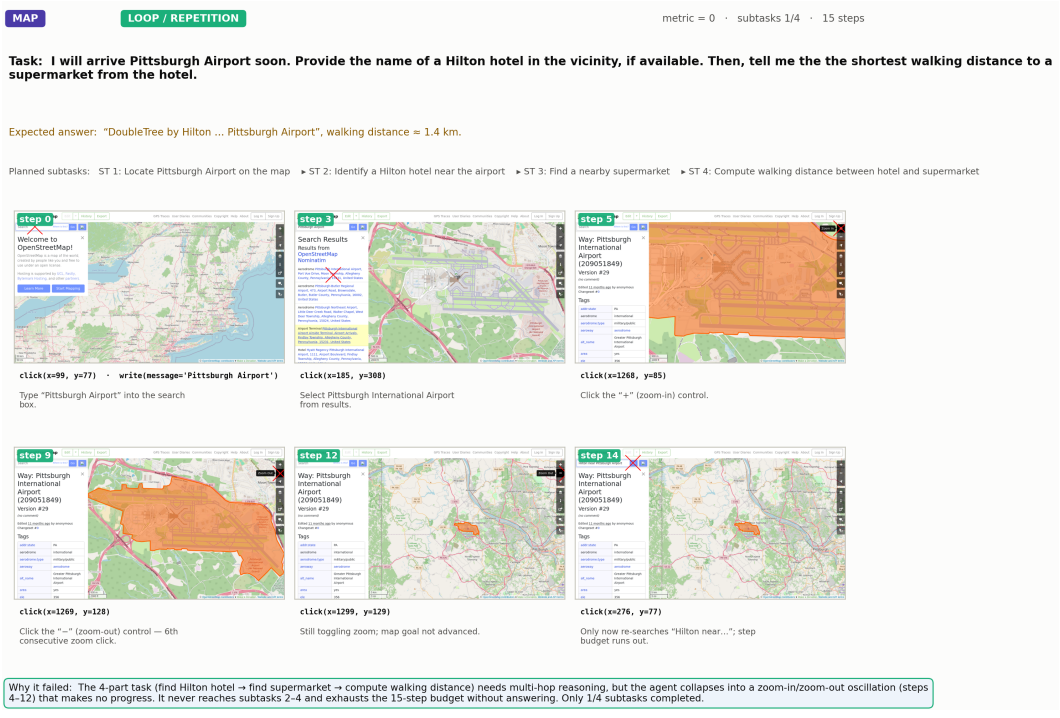



Figure 6: Failure of JHiMA-WebRL on a Map task: repeated irrelevant actions exhaust the 15-step budget.