

000 001 002 003 004 005 006 007 008 009 010 011 012 WEB AGENTS ARE STILL GREEDY: PROGRESS- 001 AWARE ACTION GENERATION AND SELECTION VIA 002 META-PLAN 003 004 005

006 **Anonymous authors**

007 Paper under double-blind review
008
009
010
011
012

ABSTRACT

013 Despite recent advancements in the reasoning and planning capabilities of large
014 language models that enable automated web agent tasks, state-of-the-art web agent
015 frameworks still exhibit high task failure rates and lag behind human performance,
016 hindering their deployment and generalization across diverse website environ-
017 ments. In this paper, we identify key limitations in these web agent frameworks,
018 attributing failure to their greedy reasoning without an understanding of the cur-
019 rent task progress regarding what key steps have been completed and what re-
020 mains in the next steps. Due to this lack of progress awareness, existing web
021 agent frameworks often fall into suboptimal behaviors such as skipping essential
022 key steps and producing incoherent or oscillatory trajectories, which hinder task
023 completion. To address these limitations, we propose MAPLE, a simple yet ef-
024 fective add-on method with **MetA-PLan** guided action generation and **sElection**.
025 Our proposed method equips existing web agent frameworks to self-reason with
026 an explicit meta-plan that encompasses high-level sequential guidelines for solv-
027 ing the task, enabling them to keep track of current progress and consistently
028 adhere to the given guidelines. Experiments across diverse website benchmarks
029 demonstrate that MAPLE largely outperforms previous state-of-the-art web agent
030 frameworks by addressing their common failures and suboptimal behaviors in-
031 duced by the lack of progress awareness.
032
033

1 INTRODUCTION

034 Recent advancements in the reasoning capabilities of Large Language Models (LLMs) have shown
035 versatility in a wide range of complex language reasoning tasks (Wei et al., 2022; Wang et al.,
036 2022b; Liu et al., 2024), as well as agentic applications (Yao et al., 2023; Shinn et al., 2023) that
037 require sequential interaction with environments. Among these, however, generalization on web
038 agent tasks (Zhou et al., 2023; Koh et al., 2024a; Deng et al., 2023) still remains limited and lags
039 behind human performance, due to the complexity of long-horizon planning and the dynamic nature
040 of web environments which require interacting with a large number of graphical user interface (GUI)
041 elements such as buttons and input fields. To better generalize on such complex website environ-
042 ments, several web agent frameworks (Koh et al., 2024b; Gu et al., 2024) have focused on adaptive
043 control and planning strategies to cope with dynamically changing web observations in response to
044 each action taken by the agent. Specifically, TreeSearch (Koh et al., 2024b) proposed a trial-and-
045 error planning approach that allows agents to backtrack to previous states upon encountering failure,
046 while WebDreamer (Gu et al., 2024) adopted simulation-based planning that imagines the expected
047 outcomes of each plan and selects the one most aligned with the desired final goal.

048 However, these state-of-the-art (SOTA) frameworks still exhibit limited generalization on complex
049 long-horizon tasks that require structured reasoning. In this paper, we identify the key limitations
050 of these frameworks induced by their greedy reasoning (Yao et al., 2023) to predict the next action,
051 without an understanding of the current progress regarding what key steps have been completed and
052 what remains in the next steps. Due to this lack of progress awareness, they exhibit suboptimal
053 behaviors such as skipping essential key steps and producing incoherent action trajectories. As
illustrated in Figure 1, the agent greedily predicts next actions to achieve the goal without proper
grounding on current progress, resulting in incoherent behavior such as scrolling down the page

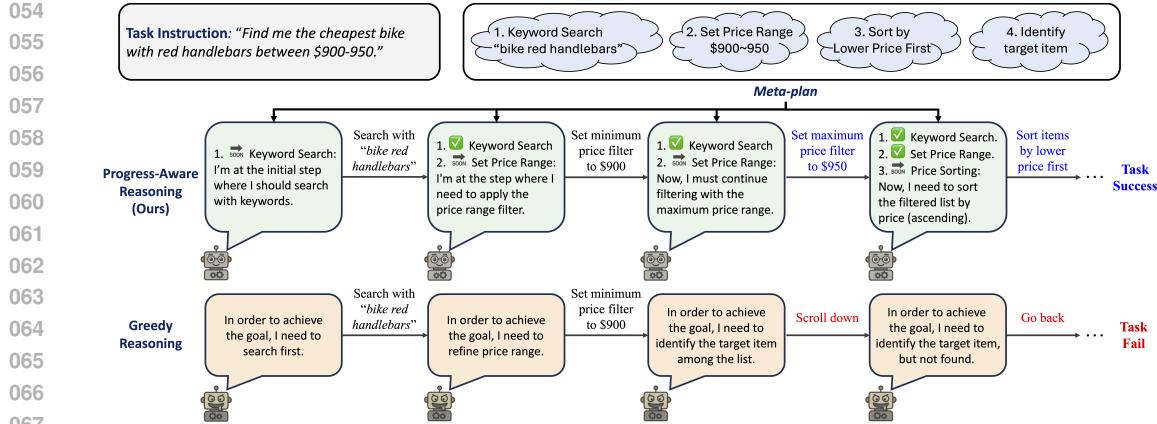


Figure 1: Agent trajectories under different reasoning approaches. Upper flow: progress-aware reasoning approach (ours). Lower flow: greedy reasoning (Gu et al., 2024; Koh et al., 2024b). Progress-aware reasoning via the meta-plan enables the agent to successfully complete the task by maintaining a coherent trajectory, without skipping essential steps.

where it is expected to coherently set the maximum price, and further skipping the essential sorting step required to identify the cheapest item.

To address these limitations, we propose MAPLE, which incorporates web agents with **MetA-PLan** guided action generation and sElection to enable more faithful and coherent navigation. Specifically, we first establish a meta-plan where the given task instruction is decomposed into the high-level subtasks that are essential for achieving the goal. At each sequential step, the agent predicts the next action based on understanding of the current progress, specifically reasoning about which subtasks have been completed so far and which remain within the meta-plan. As shown in Figure 1 and Appendix E, this simple yet effective progress-aware reasoning addresses the common failures of SOTA web agents (Gu et al., 2024; Koh et al., 2024b), such as incoherent or oscillatory trajectories, premature termination, and invalid action repetition, by explicitly following the meta-plan that helps steer toward the given task goal.

Moreover, we note that such progress-aware reasoning can also address a key limitation of the action value function from SOTA web agents (Gu et al., 2024; Koh et al., 2024b), which quantifies the value of each action candidate to select the most promising one that can be taken on the current web observation. Specifically, conventional value functions tend to assign the same scores to all *on-track* action candidates that can plausibly serve as intermediate steps toward the goal, failing to identify a more goal-directed one from less effective yet valid on-track ones. For example, in tasks that involve finding the most recent item from the shopping listings, they may select inefficient, detouring on-track action (e.g., jumping to the oldest page and searching backwards) rather than the effective, straightforward action (e.g., searching from the latest page). This inefficiency hampers the agent’s progress, causing unnecessary detours that impede convergence toward the objective. To address this, we devise a new action selection method that tracks current progress and explicitly votes for the most promising next action that aligns with the next remaining step within the meta-plan. Consequently, the agent is directed toward the objective by taking more effective on-track actions based on the guidance from the meta-plan.

Our comprehensive experiments on various website benchmarks from VisualWebArena dataset (Koh et al., 2024a) demonstrate that this straightforward meta-plan guided action generation and selection achieves superior task success rates compared to the previous SOTA web agent frameworks (Gu et al., 2024; Koh et al., 2024b). Also, integrating our methods on top of these frameworks further yields substantial performance gains, achieving +40% relative improvement for Webdreamer (Gu et al., 2024) in Classifieds benchmark. In summary, our contribution is threefold:

- We systematically reveal the key limitations of state-of-the-art web agent frameworks, where greedy reasoning without keeping track of the current progress status leads to sub-optimal trajectories.
- We propose meta-plan guided reasoning for action generation and selection that enables web agents to explicitly track their current progress and consistently take more goal-directed actions, promoting more faithful and coherent web navigation.

108 • Extensive experiments on various website benchmarks show that our method achieves su-
 109 perior performance compared to the previous state-of-the-art web agent frameworks, and
 110 even further enhances their performance when our methods are integrated into them.

112 **2 RELATED WORK**

114 **2.1 WEB AGENTS**

116 With the advancements in the reasoning capability of LLMs (Wei et al., 2022; Zhou et al., 2022),
 117 the seminal works (Gur et al., 2023; Yao et al., 2023; Deng et al., 2023) employed LLMs as a
 118 controller to automate web tasks by navigating within the web environments (Deng et al., 2023;
 119 Zhou et al., 2023; Koh et al., 2024a). Since processing an enormous amount of textual tokens from
 120 HTML sources is resource-inefficient with low information density, Cheng et al. (2024); Zheng
 121 et al. (2024); Lin et al. (2025) improved its efficiency by leveraging a smaller number of visual patch
 122 tokens from web screenshots based on multimodal LLMs (Achiam et al., 2023; Wang et al., 2024).
 123 Another line of work (Gu et al., 2024; Koh et al., 2024b) improved the planning ability of web agents
 124 to interact with a large number of GUI components from dynamic and complex web environments.
 125 However, these state-of-the-art web agent frameworks still exhibit limited generalization on long-
 126 horizon tasks that may require reasoning about the current progress, including what steps have been
 127 completed and what next steps remain. Due to the lack of such progress awareness, they often suffer
 128 from suboptimal behaviors such as skipping essential key steps for solving the long-horizon tasks.
 129 To address this, we address this limitation by guiding the web agent with a meta-plan to keep track
 130 of its current progress, promoting a more faithful and coherent web navigation.

131 **2.2 AGENTS GUIDED BY META-PLAN**

133 To effectively solve the complex and long-horizon tasks, recent studies have focused on a plan-and-
 134 solve strategy via guidance from a meta-plan that contains high-level sequential concepts to solve
 135 the given task. Specifically, Khot et al. (2022); Zheng et al. (2023); Wang et al. (2023) simplified
 136 complex language reasoning tasks with the meta-plan by decomposing the given task into simpler
 137 language subtasks. Also, agentic frameworks such as TPTU (Ruan et al., 2023), HuggingGPT (Shen
 138 et al., 2023), and LLMCompiler (Kim et al., 2024) leveraged meta-plan to effectively coordinate
 139 diverse agentic tools, such as SQL database querying. Another line of work (Erdogan et al., 2025;
 140 Xiong et al., 2025) explicitly integrated the meta-plan into web agent frameworks where the agent’s
 141 action policy is explicitly governed by the high-level guidelines from the meta-plan. Plan-and-
 142 Act (Erdogan et al., 2025) developed a more sophisticated meta-plan by training the meta-planner on
 143 extra synthetic data and enabling it to adapt to unforeseen variations in dynamic web environments.
 144 Similarly, MPO (Xiong et al., 2025) also optimized the meta-planner via feedback from the agent
 145 during web navigation. Despite the advances, these frameworks may still suffer from potential plan-
 146 to-action grounding errors that may arise when the high-level meta-plan is incorrectly decoded into
 147 a single low-level action, as abstract concepts within the meta-plan are often misinterpreted by the
 148 policy model. Consequently, these grounding errors can lead to suboptimal trajectories. In this
 149 paper, we mitigate such errors by generating multiple candidate actions from the policy model and
 150 selecting the most promising one among the candidates via our proposed action evaluator, based on
 151 its alignment with the next intended step in the meta-plan.

152 **3 METHOD**

153 In this section, we first describe the problem formulation of existing web agents (Sec 3.1) and
 154 then introduce MAPLE, a general add-on method designed to enhance the existing web agents via
 155 progress-aware action generation and selection. Specifically, given a web-based task, we propose
 156 two key components for improving agent reliability and coherence: (i) Action Candidate Genera-
 157 tion with Meta-Plan (Sec 3.2), where the task is decomposed into several high-level guidelines that
 158 enable the agent to keep track of its current progress when predicting the next action candidates, and
 159 (ii) Action Selection with Meta-Plan (Sec 3.3), which opts for the most goal-aligned action among
 160 the candidates based on its relevance to the next intended step within the meta-plan. Through these
 161 two components, MAPLE enables web agents to follow a more structured and coherent execution
 162 path, mitigating common failures such as skipping essential steps and trajectory incoherence.

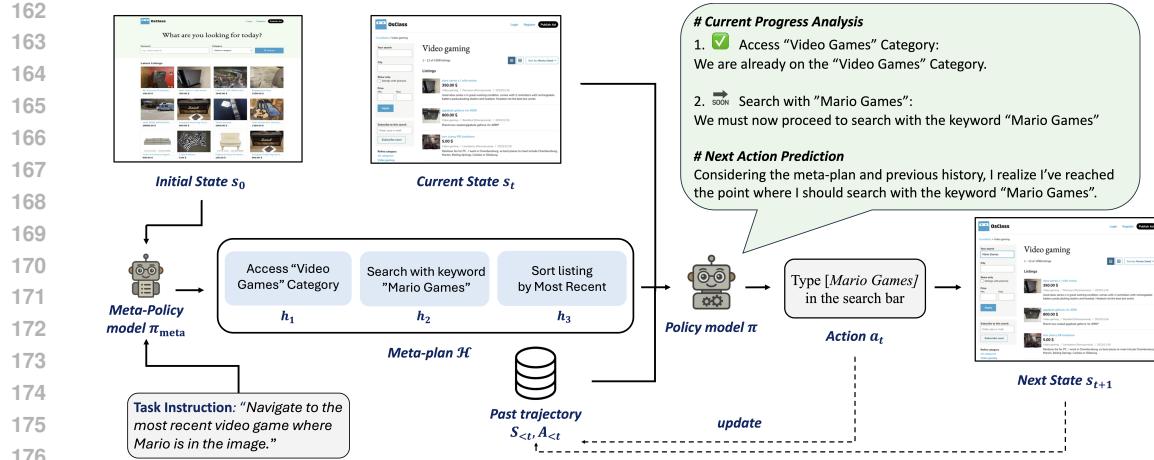


Figure 2: Overview of action generation governed by meta-plan. Based on the progress-aware reasoning, policy model generates coherent action aligned with upcoming steps in the meta-plan.

3.1 PRELIMINARY

In the standard web agent framework, an agent aims to complete a given instruction I (e.g., “*Find me the cheapest bike with red handlebars between \$900-950*”) through sequential interactions with a dynamic web environment. At each timestep t , given the current observation s_t including an HTML document and the corresponding screenshot image, the agent predicts the next action a_t (e.g., “*type ‘bike’ in the search bar*”) via a policy model π powered by multi-modal LLM:

$$a_t = \pi(s_t, I). \quad (1)$$

Under the Markov Decision Process (MDP), the policy π aims to generate an *on-track* action that transitions to the desired next state s_{t+1} , which possibly steers the agent toward the final state s_t that satisfies the objective I :

$$s_{t+1} = \mathcal{S}(s_t, a_t), \quad \text{where } \exists \hat{t} \geq t+1 \text{ s.t. } R(s_{\hat{t}} | I) = 1. \quad (2)$$

Here, \mathcal{S} denotes the website transition function and $R(\cdot)$ is a binary indicator of whether the final state $s_{\hat{t}}$ fulfills the given objective I or not. However, existing frameworks (Gu et al., 2024; Koh et al., 2024b) often fall into a suboptimal state s_{t+1}^* due to its greedy reasoning process of π . Specifically, for each time step t , the policy model π only reasons about the immediate next plan for achieving the goal under the current state s_t without explicit consideration of the past trajectory $S_{<t} = (s_0, \dots, s_{t-1})$ and $A_{<t} = (a_0, \dots, a_{t-1})$, predicting actions without adequately reflecting on what has been achieved so far and what remains in the task (Figure 1). This lack of progress-aware reasoning leads to three failure modes: 1) a suboptimal trajectory that omits essential key steps (e.g., sorting items by price for the task finding the cheapest item), 2) an incoherent trajectory that discontinues the progress achieved by prior on-track actions (e.g., scroll down action after setting minimum price, instead of coherently setting maximum price), and 3) an invalid repetitive behavior without completing the objective (e.g., repeating the scroll down action forever).

3.2 ACTION CANDIDATE GENERATION WITH META-PLAN

To promote more coherent and goal-aligned behavior, we explicitly ground the policy model π on a meta-plan $\mathcal{H} = (h_1, h_2, \dots, h_K)$, where each h_k denotes a high-level subtask that must be completed in sequence to fulfill the objective I , as illustrated in Figure 2:

$$\mathcal{H} = \pi_{\text{meta}}(I, s_0). \quad (3)$$

Here, π_{meta} denotes a meta-policy model which decomposes the objective I into a sequence of high-level subtask plans h_k starting from the initial state s_0 . In practice, we employ the off-the-shelf multi-modal LLM (Hurst et al., 2024) as the meta-policy model to generate up to $K = 5$ subtasks (See Appendix A for the full prompt and example outputs). This meta-plan serves as a guide for the agent to keep track of its own progress and instills a structured vision of remaining subgoals. To ground the agent’s next action based on current progress within the meta-plan, we revise Eq. 1

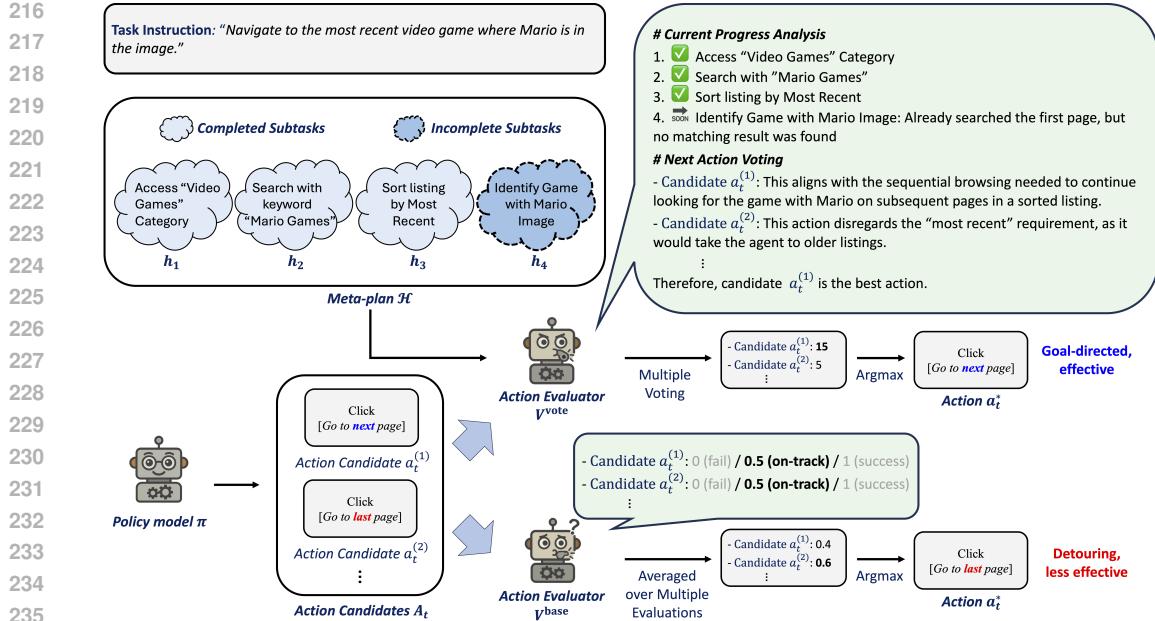


Figure 3: Overview of action selection governed by the meta-plan. Without guidance from the meta-plan (bottom), the conventional evaluator may indiscriminately select on-track yet detouring action (e.g., backward search despite the listings already being sorted by most recent first). In contrast, our evaluator (top), guided by progress-aware reasoning, explicitly votes for the effective on-track action (e.g., forward search) aligned with the next remaining subgoal in the meta-plan, thereby steering the agent onto a goal-directed trajectory.

for the policy model π to be conditioned additionally on the past trajectory $(S_{<t}, A_{<t})$, and the meta-plan \mathcal{H} :

$$a_t = \pi(s_t, I | S_{<t}, A_{<t}, \mathcal{H}). \quad (4)$$

Under this reformulation, the policy model π first analyzes current progress within the meta-plan by inspecting the past trajectory, and then predicts the next corresponding action candidate to achieve the remaining incomplete subtasks within the meta-plan. Specifically, for the policy model to self-reason on the current progress at each time step t , we define a binary completion variable $c_k \in \{0, 1\}$ for each subtask h_k , resulting in a progress vector $\mathbf{c}_t = [c_1, c_2, \dots, c_K]$ where each $c_k = 1$ indicates that h_k is considered completed:

$$\mathbf{c}_t = [\mathcal{C}(h_k | S_{\leq t}, A_{<t})]_{k=1}^K \quad (5)$$

where $\mathcal{C}(\cdot)$ denotes judgment from the policy model whether each subtask has been completed by inspecting the action history $A_{<t}$ and the corresponding observation history $S_{\leq t} = (s_0, \dots, s_t)$. To operationalize $\mathcal{C}(\cdot)$, we provide the policy model π with few-shot reasoning examples that demonstrate how to infer completion status c_k for each subtask h_k (See Appendix B.2 for the full reasoning example). Using \mathbf{c}_t , the policy model then identifies the next upcoming incomplete subtask h_{k^*} where $c_1 = 1, \dots, c_{k^*-1} = 1$, and $c_{k^*} = 0$. Finally, the policy model generates the next corresponding action candidate that can possibly complete h_{k^*} . By conditioning the action candidate generation process on this progress-aware context, the agent avoids falling into the suboptimal, redundant, or oscillatory trajectories and maintains a coherent trajectory to achieve the given objective (Figure 1 and Appendix E).

3.3 ACTION SELECTION WITH META-PLAN

For more reliable navigation of web agent, the policy model generates multiple action candidates available at each time step, and then the best candidate is selected via action value function $V^{\text{base}}(\cdot)$:

$$\mathcal{A}_t = \{a_t^{(1)}, \dots, a_t^{(N)}\} \sim \pi(s_t, I | S_{<t}, A_{<t}, \mathcal{H}; \tau_g), \quad (6)$$

$$a_t^* = \arg \max_{a \in \mathcal{A}_t} \frac{1}{M} \sum_{m=1}^M V^{\text{base}}(s_t, a; \tau_e) \quad (7)$$

270 where N and M denote the number of generated action candidates and action evaluation rounds,
 271 respectively, while τ_g and τ_e are temperature parameters for π and V^{base} . Although the generated
 272 candidates are considered *on-track* as they generally follow the meta-plan \mathcal{H} and thus maintain
 273 progress toward the objective, not all on-track candidates are equally effective. Some actions may
 274 advance the agent more directly toward next subgoal in the meta-plan, while others may take detours
 275 and result in slower progress. However, conventional action value function V^{base} (Gu et al., 2024;
 276 Koh et al., 2024b) evaluates each action candidate using a coarse rubric, assigning a score of 1.0 to
 277 actions that may finally complete the task and 0 to those likely to fail, while treating all the candidates
 278 deemed on-track equally with a score of 0.5 regardless of how effectively each action candidate
 279 advances the agent toward the goal. Although scores are averaged across multiple evaluations,
 280 such value functions often fail to identify the most effective on-track action, resulting in suboptimal
 281 decisions and ineffective trajectories (Figure 3). One might naively attempt to mitigate this by
 282 increasing granularity of scoring scheme to distinguish among on-track actions by effectiveness, but
 283 defining such a rubric for the action value function without a concrete criterion (e.g., “*give higher
 284 value to more effective on-track actions while lower values to less effective ones*”) can be subjective
 285 and may lead to unreliable and inconsistent evaluations. To address this, instead of relying on the
 286 quantification of each action candidate without a certain criterion, we explicitly select the most
 287 effective on-track action by majority voting (Wang et al., 2022a) for the one most aligned with the
 288 next remaining subgoal in the meta-plan, which helps progress towards achieving the final objective:

$$289 \quad a_t^* = \arg \max_{a \in \mathcal{A}_t} \sum_{m=1}^M \mathbb{I}(a = V_m^{\text{vote}}(s_t, \mathcal{A}_t \mid \mathcal{S}_{<t}, \mathcal{A}_{<t}, \mathcal{H}; \tau_e)) \quad (8)$$

291 where $\mathbb{I}(\cdot)$ denotes a binary indicator of whether the candidate a was selected by the voting model
 292 $V_m^{\text{vote}}(\cdot)$ in the m -th voting round. For the voting model V_m^{vote} , we employ the off-the-shelf multi-
 293 modal LLM (Hurst et al., 2024) by default. M and τ_e denote the number of voting rounds and
 294 the temperature for voting, respectively, with τ_e set to 1 by default to encourage diverse reasoning
 295 across voting rounds. In each round, $V_m^{\text{vote}}(\cdot)$ first identifies the next upcoming incomplete subgoal
 296 h_{k^*} where $c_1 = 1, \dots, c_{k^*-1} = 1$, and $c_{k^*} = 0$, as in Sec. 3.2. Subsequently, it analyzes all
 297 the action candidates in terms of how effectively each contributes to achieving h_{k^*} , and then votes
 298 for the most suitable one (See Appendix C for the full prompt and few-shot reasoning example).
 299 This progress-aware action selection guided by the meta-plan empowers the agent to identify a
 300 more goal-directed and effective trajectory toward completing the task. As shown in Figure 7 and
 301 Figure 8, our proposed action selection is also effective for addressing other common failure modes
 302 of the conventional value function, such as premature termination and invalid action repetition.

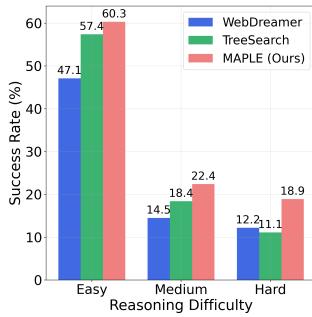
303 4 EXPERIMENTS

306 **Dataset.** We evaluate MAPLE on the widely used web agent benchmark, VisualWebArena (Koh
 307 et al., 2024a) (VWA) dataset to compare the performance against SOTA web agent frameworks.
 308 VWA consists of 910 tasks across three website environment subsets: Classifieds, Reddit, and
 309 Shopping. All the tasks are designed for evaluating the multi-modal agents, which require visual
 310 understanding of the webpage contents. Following Zheng et al. (2024); Gu et al. (2024); Koh et al.
 311 (2024b), we augmented the webpage screenshot with Set-of-Mark (Yang et al., 2023) prompting,
 312 which allows each HTML element to be grounded by its id when generating action candidates. For
 313 the evaluation metric, the success rate is calculated as the proportion of successfully completed tasks
 314 (e.g., finally navigated url matches the desired web url) to the entire number of tasks.

315 **Implementation details.** When generating action candidates for each time step, the policy π is
 316 provided with in-context examples that demonstrate how to capture current progress within the meta-
 317 plan and predict the corresponding next action (See Appendix B.2). We set the maximum number
 318 of generated action candidates $N = 20$ and use nucleus sampling (Holtzman et al., 2019) with top- p
 319 of 0.95 and temperature $\tau_g = 1.0$ in Eq. 6, unless specified. Among the generated action candidates,
 320 we feed top-5 frequently generated actions to our action selection (Eq. 8) to vote for the best action
 321 a_t^* . For the voting model V^{vote} , we provide in-context examples that demonstrate how to select the
 322 most promising action based on current progress (See Appendix C.2). Also, we set the temperature
 323 $\tau_e = 1.0$ and the number of voting rounds $M = 20$, unless specified. The agent is forced to
 324 stop execution after a maximum of 30 time steps, while treating as a failure if the same actions are

324 Table 1: Performance comparison with state-of-the-art web agent frameworks on VWA (Koh et al.,
 325 2024a). Values in the parentheses (Δ) denote relative improvement compared to the vanilla method,
 326 ReAct (Yao et al., 2023). For fair comparisons, we used GPT-4o for all the methods.
 327

328 Method	Action Gen.	Action Select	Success Rate (SR) \uparrow			Total SR (Δ) \uparrow
	w/ Meta-Plan?	w/ Meta-Plan?	Classifieds	Reddit	Shopping	
330 ReAct (Yao et al., 2023)	\times	\times	17.9	14.3	19.3	17.6 (+0.0%)
331 ICAL (Sarch et al., 2024)	\times	\times	-	-	-	23.4 (+33.0%)
332 AdaptAgent (Verma et al., 2024)	\times	\times	-	-	-	23.9 (+35.8%)
333 GenericAgent (Chezelles et al., 2024)	\times	\times	-	-	-	26.7 (+51.7%)
334 WebDreamer (Gu et al., 2024)	\times	\times	23.2	17.5	26.3	23.2 (+31.8%)
335 TreeSearch (Koh et al., 2024b)	\times	\times	26.8	20.6	28.9	26.2 (+48.9%)
336 MAPLE (Eq 6, 7)	✓	\times	29.9	24.8	33.9	30.8 (+75.0%)
337 MAPLE (Eq 6, 8)	✓	✓	32.1	26.7	36.9	33.3 (+89.2%)



349 Figure 4: Comparison of task
 350 success rates depending on
 351 reasoning difficulty level in Classi-
 352 fieds subset of VWA.

339 Table 2: Add-on effect of the proposed methods on SOTA frame-
 340 works, WebDreamer and TreeSearch. Values in the parentheses
 341 (Δ) denote the relative improvement compared to each baseline
 342 framework.

Method	Action Gen. w/ Meta-Plan?	Action Select w/ Meta-Plan?	Classifieds SR (Δ) \uparrow
WebDreamer (Gu et al., 2024)	\times	\times	23.2 (+0.0%)
+ MAPLE (Eq 6, 7)	✓	\times	29.9 (+28.9%)
+ MAPLE (Eq 6, 8)	✓	✓	32.5 (+40.1%)
TreeSearch (Koh et al., 2024b)	\times	\times	26.8 (+0.0%)
+ MAPLE (Eq 6, 7)	✓	\times	28.6 (+6.7%)
+ MAPLE (Eq 6, 8)	✓	✓	29.9 (+11.6%)

354 repeated for 5 consecutive steps (Gu et al., 2024). For all experiments, we employed GPT-4o (Hurst
 355 et al., 2024) for the policy model π , the meta-policy model π_{meta} , and the action evaluator V^{vote} .
 356

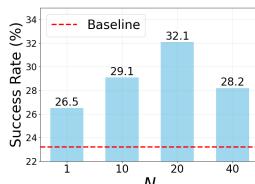
358 4.1 MAIN RESULTS

360 **Comparison with SOTA frameworks.** In Table 1, we compare MAPLE with other state-of-the-
 361 art web agent frameworks on VWA benchmark. The vanilla method, ReAct, shows a low success
 362 rate across all the subsets due to its greedy reasoning without tracking current progress during the
 363 action generation process, while recent baselines enhanced the performance via human feedback
 364 (ICAL and AdaptAgent) and sophisticated planning strategies with simulation- or tree-based explo-
 365 ration (WebDreamer and TreeSearch). However, MAPLE consistently exhibits the best success rates
 366 on all the subsets, largely outperforming all the baselines. Specifically, when the action generation
 367 process is guided by the meta-plan as in Eq. 6, it already outperforms the previous SOTA GenericA-
 368 gent by a large margin, achieving +4.1% absolute points in total. Also, applying the action selection
 369 via meta-plan (Eq 8) brings further improvement of +6.6% absolute points over GenericAgent, while
 achieving +89% relative improvement compared to the vanilla method.

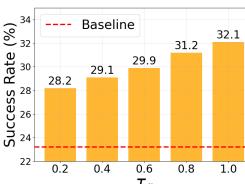
370 In Figure 4, we further analyze the performance across different levels of reasoning difficulty. No-
 371 tably, the baselines such as WebDreamer and TreeSearch substantially degrade their performance
 372 as the reasoning difficulty increases. Due to the greedy reasoning that lacks awareness of current
 373 progress, they exhibit suboptimal behavior such as skipping essential key steps in the hard tasks that
 374 require long-horizon planning (Figure 1). However, MAPLE shows significantly improved perfor-
 375 mance on hard tasks, achieving +70% relative improvement compared to TreeSearch. These results
 376 demonstrate the effectiveness of our progress-aware reasoning for action generation and selection
 377 via meta-plan, by steering away from falling into the suboptimal trajectories and instead pursuing a
 more reliable and coherent trajectory for completing the long-horizon tasks.

378
 379 Table 3: Ablation study of progress-aware rea-
 380 soning (PAR) and meta-plan for action genera-
 381 tion, evaluated on Classifieds depending on the
 382 level of reasoning difficulty. Values in the paren-
 383 thesies (Δ) denote the relative overall im-
 384 prove-
 385 ment compared to the baseline method (Gu
 386 et al., 2024) without progress-aware reason-
 387 ing.

PAR?	PAR w/ Meta-plan?	Reasoning difficulty (SR) \uparrow			Classifieds SR (Δ) \uparrow
		Easy	Medium	Hard	
\times	\times	47.1	14.5	12.2	23.2 (+0.0%)
\checkmark	\times	51.5	18.4	15.6	26.9 (+15.9%)
\checkmark	\checkmark	55.9	22.4	16.7	29.9 (+28.9%)

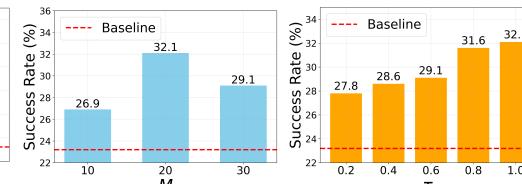


397 (a) N candidates
 398 ($\tau_g = 1.0$)

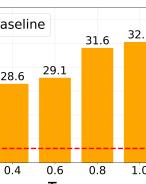


397 (b) Temperature τ_g
 398 ($N = 20$)

Action Gen. w/ Meta-Plan?	Dynamic Replanning?	Reasoning difficulty (SR) \uparrow			Classifieds SR (Δ) \uparrow
		Easy	Medium	Hard	
\times	\times	47.1	14.5	12.2	23.2 (+0.0%)
\checkmark	\times	55.9	22.4	16.7	29.9 (+28.9%)
\checkmark	\checkmark	55.9	21.1	14.4	28.6 (+23.3%)



397 (c) M voting
 398 ($\tau_e = 1.0$)



397 (d) Temperature τ_e
 398 ($M = 20$)

400 Figure 5: Ablation study on hyperparameters in meta-plan guided action generation and selection.
 401 (a), (b): Effects of varying the number of action candidates N and temperature τ_g in action genera-
 402 tion, respectively. (c), (d): Effects of varying the number of voting rounds M and temperature τ_e in
 403 action selection, respectively. Red dotted lines indicate the performance of the baseline method (Gu
 404 et al., 2024) without meta-plan guidance.

405 **Plug-and-play with SOTA frameworks.** Since our proposed methods can be easily integrated
 406 into any prompt-based web agent frameworks via few-shot reasoning examples, we further investi-
 407 gate the add-on effect of MAPLE into the state-of-the-art frameworks, WebDreamer and TreeSearch.
 408 In Table 2, both action generation and selection via meta-plan consistently improve the success rates
 409 of all the baseline frameworks, yielding cumulative performance gains. Specifically, our action gen-
 410 eration brings a remarkable +29% relative improvement to WebDreamer, while our action selection
 411 further contributes an additional +40% relative improvement. These results further corroborate that
 412 MAPLE is not only effective as a standalone framework but also serves as a generalizable plug-
 413 and-play module that resolves common failure modes (Appendix E) in conventional frameworks,
 414 thereby yielding substantial performance gains.

4.2 ABLATION STUDIES AND ANALYSIS

415 We conduct ablation studies and detailed analysis to assess the effectiveness of each component and
 416 the sensitivity of the hyperparameters comprising MAPLE. Unless specified, we applied both meta-
 417 plan guided action generation (Eq. 6) and selection (Eq. 8). Also, we report task success rate (SR)
 418 on Classifieds subset in VWA benchmark (Koh et al., 2024a).

419 **Effect of progress-aware reasoning.** In Table 3, we analyze the effect of progress-aware reason-
 420 ing (Eq. 5) when generating next action candidates. Interestingly, equipping the policy model π with
 421 the ability to naively reason about the agent’s current progress by only analyzing history of actions
 422 $A_{\leq t}$ and observations $S_{\leq t}$, even without the meta-plan \mathcal{H} , substantially improves the performance
 423 over the vanilla greedy action generation that lacks such progress-aware reasoning. Moreover, when
 424 the policy model π is further conditioned on the meta-plan to reason about the current progress and
 425 to generate next actions coherent with the remaining steps of the meta-plan (Sec 3.2), the success rate
 426 is additionally enhanced, up to 29% of relative improvement from the baseline. We also observe that
 427 our progress-aware reasoning is effective in resolving challenging failure cases that involve falling
 428 into incoherent or oscillatory trajectories, as shown in Figure 1 (See Appendix E for more qual-
 429 itative examples). These results highlight that the progress awareness via meta-plan is essential for
 430 promoting web agents to navigate under a coherent and goal-directed trajectory.

432 **Effect of N and τ_g for action generation.** For generating action candidates that adhere to the
 433 meta-plan as in Eq. 6, we investigate the effect of the number of generated action candidates N
 434 and the temperature τ_g when decoding each candidate. In Figure 5a, for all the choices of N , our
 435 proposed action generation consistently brings improved performance compared to the baseline case
 436 without meta-plan guidance. We observe that the success rate increases monotonically as N grows
 437 up to 20, while further increasing N up to 40 only entails $2\times$ increased computation of output
 438 tokens without commensurate performance gains. Therefore, we fix $N = 20$ as a default to ensure
 439 a solid success rate while maintaining computational efficiency. We also note that leveraging only
 440 a single action candidate (i.e., $N = 1$) as in Erdogan et al. (2025), which obviates the need for the
 441 action selection process (Sec. 3.3), leaves substantial room for improvement compared to $N > 1$
 442 cases where the proposed action selection is applied. This result possibly suggests that relying on a
 443 single decoded action may not be robust to potential errors in plan-to-act grounding, as the abstract
 444 concepts in the meta-plan are often misinterpreted by the action generator. We mitigate such errors
 445 by generating multiple action candidates and introducing another verifier (Eq. 8) that selects the most
 446 promising action based on the alignment with the meta-plan. In Figure 5b, we observe a similar trend
 447 where all the choices of the temperature τ_g consistently exhibit improved performance compared to
 448 the baseline. Notably, the success rate gradually increases as τ_g grows up to 1, highlighting the
 449 importance of promoting reasoning diversity for broadening the set of promising action candidates.

450 **Effect of M and τ_e for action selection.** To identify the most promising action candidate that
 451 adheres to the meta-plan as in Eq. 8, we investigate the effect of the number of voting rounds M
 452 and the temperature τ_e in Figure 5c and 5d, respectively. Notably, our proposed action selection
 453 consistently delivers performance improvement to the baseline case without meta-plan guidance
 454 during action selection, regardless of the choices of both M and τ_e . Similar to the trend shown in the
 455 parameter study of action generation (Figure 5a, 5b), we observe that the performance is saturated
 456 at $M = 20$ to achieve the best performance. Moreover, increasing the temperature τ_e monotonically
 457 improves the success rate. This result further corroborates the importance of incorporating diverse
 458 perspectives in the voting process to identify the most promising action aligned with the meta-plan.

459 **Dynamic replanning of meta-plan.** To cope with unexpected variations during web navigation,
 460 Erdogan et al. (2025) proposed dynamically updating the meta-plan rather than relying on a static
 461 one established at the initial state. Following Erdogan et al. (2025), we further investigated the
 462 effect of dynamic replanning by updating the meta-plan after every action execution and generating
 463 subsequent actions based on the revised meta-plan. For updating the meta-plan, we prompted the
 464 meta-policy model to revise or remove parts of the remaining incomplete plan that are no longer
 465 executable for completing the task (See Appendix D for the full prompt). The results in Table 4 show
 466 that action generation guided by meta-plan consistently exhibits improved performance compared to
 467 the baseline, while further applying dynamic replanning leads to a slight performance degradation on
 468 medium- and hard-level reasoning tasks. This result possibly indicates that frequent updates of the
 469 meta-plan can rather lead to the suboptimal trajectories in the long-horizon tasks, such as generating
 470 incoherent actions that disrupt the continuity of the progress achieved by the previous meta-plan, or
 471 skipping essential steps due to premature removal of necessary plans (Figure 1).

472 5 CONCLUSION

473 In this paper, we identify the key limitations of existing state-of-the-art web agent frameworks where
 474 the greedy reasoning without progress awareness often leads to suboptimal behaviors, such as skip-
 475 ping essential procedures and generating incoherent trajectories, hampering the successful comple-
 476 tion of the web tasks. To address this, we propose the progress-aware reasoning approach to align
 477 the next action with a meta-plan that encompasses a sequence of high-level guidelines, allowing the
 478 agent to monitor its current progress and consistently adhere to the given guidelines. Also, we fur-
 479 ther revisit the fundamental limitations of existing action evaluation mechanisms where suboptimal
 480 actions are often assigned higher values and hence selected. To remedy this, we introduce a new ac-
 481 tion evaluation method by majority voting based on the meta-plan, enabling the agent to consistently
 482 select more effective actions that direct toward the goal without falling into detouring trajectories.
 483 Through extensive experiments across diverse web benchmarks and rigorous ablation studies, we
 484 demonstrate that our progress-aware action generation and selection methods can resolve the com-
 485 mon failure cases frequently encountered by the conventional web agent frameworks, achieving the
 486 state-of-the-art performance while serving as an effective plug-in module.

486 REFERENCES
487

488 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Ale-
489 man, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical
490 report. [arXiv preprint arXiv:2303.08774](https://arxiv.org/abs/2303.08774), 2023.

491 Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiy-
492 ong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. [arXiv preprint
493 arXiv:2401.10935](https://arxiv.org/abs/2401.10935), 2024.

494 De Chezelles, Thibault Le Sellier, Maxime Gasse, Alexandre Lacoste, Alexandre Drouin, Massimo
495 Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, et al. The browsergym ecosystem
496 for web agent research. [arXiv preprint arXiv:2412.05467](https://arxiv.org/abs/2412.05467), 2024.

497

498 Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su.
499 Mind2web: Towards a generalist agent for the web. [Advances in Neural Information Processing
500 Systems](https://proceedings.neurips.cc/paper/2023/file/28091-28114.pdf), 36:28091–28114, 2023.

501 Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anu-
502 manchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for
503 long-horizon tasks. [arXiv preprint arXiv:2503.09572](https://arxiv.org/abs/2503.09572), 2025.

504

505 Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari
506 Srivastava, Yanan Xie, Peng Qi, et al. Is your llm secretly a world model of the internet? model-
507 based planning for web agents. [arXiv preprint arXiv:2411.06559](https://arxiv.org/abs/2411.06559), 2024.

508 Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and
509 Aleksandra Faust. A real-world webagent with planning, long context understanding, and pro-
510 gram synthesis. [arXiv preprint arXiv:2307.12856](https://arxiv.org/abs/2307.12856), 2023.

511 Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text
512 degeneration. [arXiv preprint arXiv:1904.09751](https://arxiv.org/abs/1904.09751), 2019.

513

514 Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Os-
515 trow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. [arXiv preprint
516 arXiv:2410.21276](https://arxiv.org/abs/2410.21276), 2024.

517 Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish
518 Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. [arXiv
519 preprint arXiv:2210.02406](https://arxiv.org/abs/2210.02406), 2022.

520

521 Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W Mahoney, Kurt Keutzer,
522 and Amir Gholami. An llm compiler for parallel function calling. In [Forty-first International
523 Conference on Machine Learning](https://proceedings.mlr.press/v160/kim24a.html), 2024.

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

527

528 Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language
529 model agents. [arXiv preprint arXiv:2407.01476](https://arxiv.org/abs/2407.01476), 2024b.

530

531 Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian
532 Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui
533 visual agent. In [Proceedings of the Computer Vision and Pattern Recognition Conference](https://proceedings.cvpr.org/index.html), pp.
19498–19508, 2025.

534

535 Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao,
536 Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. [arXiv preprint
537 arXiv:2412.19437](https://arxiv.org/abs/2412.19437), 2024.

538 Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi,
539 Hangyu Mao, Ziyue Li, Xingyu Zeng, et al. Tptu: large language model-based ai agents for task
planning and tool usage. [arXiv preprint arXiv:2308.03427](https://arxiv.org/abs/2308.03427), 2023.

540 Gabriel Sarch, Lawrence Jang, Michael Tarr, William W Cohen, Kenneth Marino, and Katerina
 541 Fragkiadaki. Vlm agents generate their own memories: Distilling experience into embodied
 542 programs of thought. *Advances in Neural Information Processing Systems*, 37:75942–75985,
 543 2024.

544 Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueling Zhuang. Hugging-
 545 gpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information*
 546 *Processing Systems*, 36:38154–38180, 2023.

547 Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion:
 548 Language agents with verbal reinforcement learning. *Advances in Neural Information Processing*
 549 *Systems*, 36:8634–8652, 2023.

550 Gaurav Verma, Rachneet Kaur, Nishan Srishankar, Zhen Zeng, Tucker Balch, and Manuela Veloso.
 551 Adaptagent: Adapting multimodal web agents with few-shot learning from human demonstra-
 552 tions. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*,
 553 2024. URL <https://openreview.net/forum?id=xcPeSftIGy>.

554 Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim.
 555 Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language
 556 models. *arXiv preprint arXiv:2305.04091*, 2023.

557 Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu,
 558 Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the
 559 world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

560 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
 561 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models.
 562 *arXiv preprint arXiv:2203.11171*, 2022a.

563 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
 564 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models.
 565 *arXiv preprint arXiv:2203.11171*, 2022b.

566 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny
 567 Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in*
 568 *neural information processing systems*, 35:24824–24837, 2022.

569 Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li.
 570 Mpo: Boosting llm agents with meta plan optimization. *arXiv preprint arXiv:2503.02682*, 2025.

571 Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark
 572 prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*,
 573 2023.

574 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.
 575 React: Synergizing reasoning and acting in language models. In *International Conference on*
 576 *Learning Representations (ICLR)*, 2023.

577 Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web
 578 agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

579 Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le,
 580 and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models.
 581 *arXiv preprint arXiv:2310.06117*, 2023.

582 Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuur-
 583 mans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex
 584 reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

585 Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng,
 586 Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for build-
 587 ing autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

594 APPENDIX

595

596

597

A META-PLAN GENERATION

598

599 In this section, we provide the full prompt for generating the meta-plan in Sec 3.2 as below:

600

601

System Prompt

602

603 You are an expert web navigation planner. Given a user's task intent, your job is to break
604 down the task into a sequence of essential stages that must be completed to achieve the task.
605 These stages will be used to evaluate an agent's progress.

606

607 For each stage, provide:

608

609 1. A short but descriptive name for the stage

610

611 2. A detailed description of what should be accomplished in this stage

612

613 The stages should form a logical progression from start to completion of the task.

614

615 The number of stages should be adjusted according to the complexity of the task, typically
616 ranging from 3 to 5 stages.

617

618 Format your response as a JSON array of stage objects without additional commentary.

619

620 Example:

621

622 [
623 {
624 "stage_name": "Search",
625 "description": "Search for the target product
626 using appropriate keywords"
627 },
628 {
629 "stage_name": "Price Comparison",
630 "description": "Compare prices, potentially
631 using sorting or filtering functionality"
632 },
633 ...
634]

635

636

Current Inputs

637

638 Current webscreen page screenshot: [INITIAL WEBPAGE SCREENSHOT IMAGE]

639

640 Define the essential sequence of stages for this web task: [TASK INSTRUCTION]

641

642

643 The example generated meta-plans are given as follows. We reformatted JSON array for visibility:

644

645

TASK: Find me the cheapest bike with red handlebars between \$900-950

646

647

1. Select Category — Click on the 'Bikes' category from the category section.
2. Keyword Search — Enter 'red handlebars' into the 'Keyword' search box.
3. Set Price Range — Set the price range to \$900-\$950 using any available price filter options.
4. Search for Bikes — Click on the 'Search' button to apply the filters and search for bikes with red handlebars within the specified price range.

648
649
650

5. Sort by Price — Sort the search results by price in ascending order to find the cheapest bike.

651

652
653

TASK: Find the most recently listed coffee maker with a touch screen. Add a 5 star rating with title “Great item” and text “Would recommend!”

654

655
656
657

1. Search for Coffee Maker — Enter ‘coffee maker’ into the keyword search bar and select the ‘Appliances’ category from the dropdown menu. Then click the ‘Search’ button.
2. Sort by Latest Listings — Once the search results are displayed, sort them by the most recently listed items to find the newest coffee maker.
3. Select Coffee Maker with Touch Screen — Browse the sorted listings to find a coffee maker that specifically mentions having a touch screen in its description, title, or image, and then click on it to view the details.
4. Add Rating and Review — On the coffee maker’s detail page, locate the section for adding a review. Provide a 5-star rating, and enter the title ‘Great item’ and the text ‘Would recommend!’ for your review.
5. Submit Review — Submit your review to finalize the rating and feedback for the coffee maker.

658

659
660

661

662
663
664

665

666
667
668669
670
671

672

673 B ACTION CANDIDATE GENERATION

674

675 In this section, we provide the full prompt with few-shot reasoning example for our proposed action
676 candidate generation in Sec 3.2.

677

678

679 B.1 SYSTEM PROMPT

680

681 System Prompt

682
683
684

685 You are an autonomous intelligent agent tasked with navigating a web browser. You will be
686 given web-based tasks. These tasks will be accomplished through the use of specific actions
687 you can issue.

688 Here’s the information you’ll have:

689

690

691
692

693

694
695

696

697
698

699

700
701

- **User’s objective:** This is the task you’re trying to complete.
- **Current URL:** This is the page you’re currently navigating.
- **Current screenshot:** A screenshot of the current webpage, with each interactable element assigned a unique numerical id.
- **Current Observation:** Lists the IDs of all interactable elements in the format [id] [tagType] [text content]. tagType is the type of the element, such as button, link, or textbox.
- **Open tabs:** The tabs you have opened.
- **Previous actions:** History of actions that you have performed. It may be helpful to track your progress.
- **Previous screenshots:** History of previous screenshots leading up to the current state by previous actions history. It may be helpful to track your progress.
- **Meta-plan:** A structured high-level guideline outlining the logical sequence of steps you should consistently follow to effectively solve any given task.

702 Actions you can perform:

702

- 703 • *Page Operations*: click [id], type [id] [content], hover [id],
704 scroll [down], etc.
- 705 • *Tab Management*: new_tab, tab_focus [index], close_tab
- 706 • *Navigation*: goto [url], go_back, go_forward
- 707 • *Completion*: stop [answer]

708

709 **Rules to Follow:**

710 1. Only issue valid action from the current observation.

711 2. Issue one action at a time.

712 3. Follow the examples to reason step by step and then issue the next action.

713 4. Based on the meta-plan, explicitly identify your current stage by analyzing previous
714 history and the current observation. Then, reason clearly about the immediate next
715 step required to complete the next unfinished stage in the meta-plan, as part of a
716 coherent plan to ultimately achieve the objective.

717 5. Output action in the form: In summary, the next action I will
718 perform is 'click [1234]'.
719

720 6. Use stop [answer] when you think the task is completed.

721

722 **B.2 FEW-SHOT REASONING EXAMPLES**

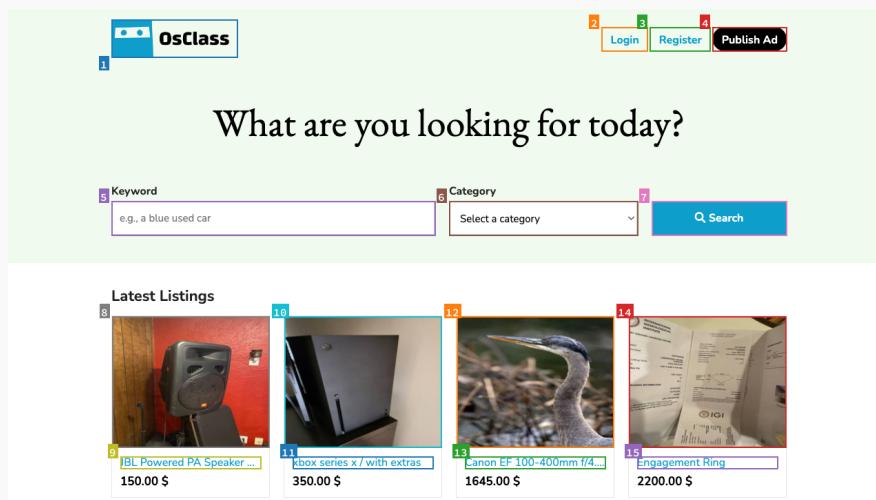
723 **Few-shot Reasoning Examples**

724 **## Inputs**

725 **Objective:** Help me find the cheapest dark colored guitar.

726 **Current URL:** http://_CLASSIFIEDS_DOMAIN_

727 **Current screenshot:**



746 **Current Observation:**

747

748 [...]

749 [5] [INPUT] []

750 [6] [SELECT] [Select a category]

751 [7] [BUTTON] [Search]

752 [9] [A] [JBL Powered PA Speaker w/ Stand]

753 [11] [A] [xbox series x / with extras]

754 [...]

755 **Open tabs:** [Classifieds]

756

757

758

759

Previous actions: None
Previous screenshots: None
Meta-plan:

760

761

762

763

764

765

766

767

768

769

770

771

772

773

- 1. Navigate to classifieds website: Access the classifieds website where musical instruments are sold.
- 2. Search for guitar listings: Use the search feature by entering ‘guitar’ to display relevant listings.
- 3. Sort by price: Sort the guitar listings in ascending order of price.
- 4. Filter by Color and Price: Review the sorted guitar listings to identify guitars that are dark-colored and note their prices.
- 5. Select Cheapest Guitar: Compare identified dark-colored guitars and choose the listing with the lowest price.

Output

Reasoning:

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

B.3 CURRENT INPUTS

793

Current Inputs

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

Objective: [TASK INSTRUCTION]
Current URL: [CURRENT URL]
Current screenshot: [CURRENT WEBPAGE SCREENSHOT IMAGE]
Current Observation: [CURRENT OBSERVATION]
Open tabs: [OPEN TABS]
Previous actions: [PREVIOUS ACTIONS]
Previous screenshots: [PREVIOUS SCREENSHOTS]
Meta-plan: [META-PLAN]

C ACTION SELECTION

In this section, we provide the full prompt with few-shot reasoning example for our proposed action selection in Sec 3.3.

810 C.1 SYSTEM PROMPT
811812 **System Prompt**
813814 You are an expert in evaluating the performance of a web navigation agent. The agent is
815 designed to help a human user navigate a website to complete a task. Given a user's task,
816 a meta-plan to complete the task (i.e., the ideal step-by-step guidelines a successful agent
817 should follow to complete the task), current webpage state, and multiple candidate actions
818 that can be taken at current state, your job is to select the best action that would most effec-
819 tively help complete the task.
820821 Here's the information you'll have:
822823

- 824 • **User's objective:** This is the task you're trying to complete.
825
- 826 • **Current URL:** This is the page you're currently navigating.
827
- 828 • **Current screenshot:** A screenshot of the current webpage, with each interactable
829 element assigned a unique numerical id.
830
- 831 • **Current Observation:** Lists the IDs of all interactable elements in the format [id]
832 [tagType] [text content]. tagType is the type of the element, such as
833 button, link, or textbox.
834
- 835 • **Open tabs:** The tabs you have opened.
836
- 837 • **Previous actions:** History of actions that you have performed. It may be helpful to
838 track your progress.
839
- 840 • **Previous screenshots:** History of previous screenshots leading up to the current
841 state by previous actions history. It may be helpful to track your progress.
842
- 843 • **Meta-plan:** A structured high-level guideline outlining the logical sequence of
844 steps you should consistently follow to effectively solve any given task.
845
- 846 • **Target actions:** The set of candidate target actions that can be performed at the
847 current webpage state.
848

849 **The types of target actions:**
850851

- 852 • *Page Operations:* click [id], type [id] [content], hover [id],
853 scroll [down], etc.
854
- 855 • *Tab Management:* new_tab, tab_focus [index], close_tab
856
- 857 • *Navigation:* goto [url], go_back, go_forward
858
- 859 • *Completion:* stop [answer]
860

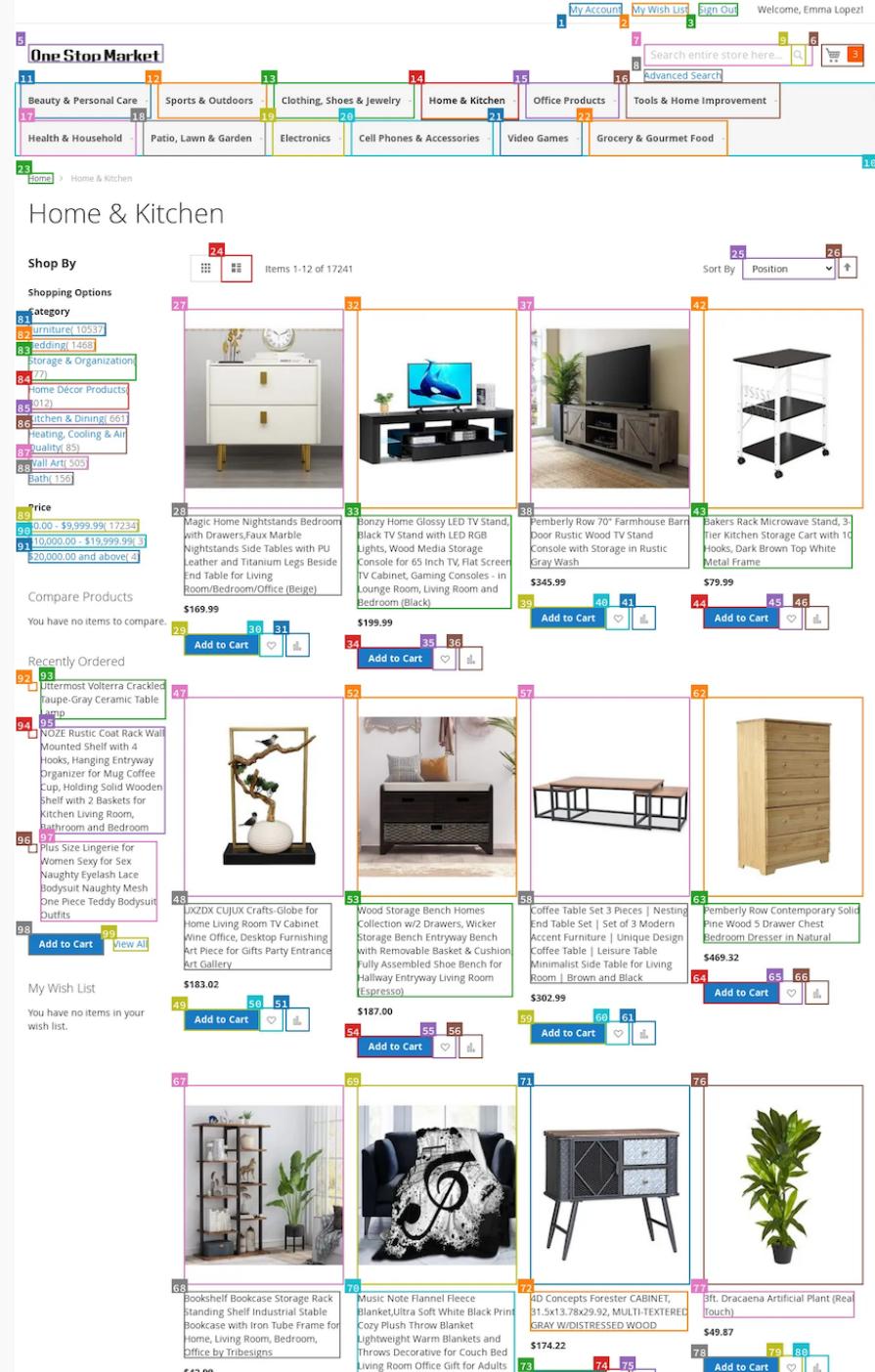
861 **Rules to Follow:**
862863

- 864 1. Given multiple candidate actions that could be taken at the current state, your job
865 is to compare all the action candidates and select the best one that would most effec-
866 tively move the user toward completing the task, specifically by considering how
867 far it would move the user forward along the meta-plan (i.e., the ideal trajectory).
868
- 869 2. Follow the examples to reason step by step and then select the best action.
870
- 871 3. You should provide reasoning that explains why the selected action is better than
872 the other candidates, considering the user's task, the meta-plan, and the history of
873 actions and screenshots.
874

856 C.2 FEW-SHOT REASONING EXAMPLES
857858 **Few-shot Reasoning Examples**
859860 **## Inputs**
861862 **Objective:** Buy the least expensive red blanket (in any size) from "Blankets & Throws"
863 category.
864

864
865
866

Current URL: `http://_SHOPPING_DOMAIN_/_home_kitchen.html`
Current screenshot:

867
868
869
870
871
872873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

918
919
920
921
922
923

[...]

Open tabs: [Shopping]**Previous actions:** None**Previous screenshots:** None**Meta-plan:**924
925
926
927
928
929
930
931
932
933
934

- 1. Navigate to Shopping Website: Open the shopping website where the purchase will be made.
- 2. Access 'Blankets & Throws' Category: Navigate to the 'Blankets & Throws' category from the homepage.
- 3. Filter and sort: Apply a filter to show only red blankets (if available) and sort the results by price in ascending order.
- 4. Select Least Expensive Blanket: Choose the least expensive red blanket from the sorted list.
- 5. Add to Cart and Purchase: Add the selected blanket to the cart and proceed to complete the purchase by providing necessary payment and shipping information.

Target actions:935
936
937
938
939

- 1. 'click [82]' where [82] is [A] element with content [Bedding (1468items)]
- 2. 'type [7] [blankets & throws\n]' where [7] is [INPUT] element with content []

Output

940
941
942
943**Reasoning:**944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

- Step 1 (Task Analysis):* The user's task is to buy the least expensive red blanket (any size) from the 'Blankets & Throws' category.

- Step 2 (Current Progress Analysis):*

- ✓ 1. Navigate to Shopping Website — The user is already on the homepage.
- ✗ 2. Access 'Blankets & Throws' Category — This is the next required step.

The agent must first access the appropriate product category ('Blankets & Throws') before applying color filters or sorting by price.

- Step 3 (Action Candidate Evaluation):*

- Candidate 0: `click [82]` — Clicks on 'Bedding (1468 items)'
 - ✓ Most logical next step, as 'Blankets & Throws' would likely be a subcategory within 'Bedding'.
 - ✓ Aligns directly with the second step of the meta-plan.
- Candidate 1: `type [7] [blankets & throws\n]`
 - ✗ Search may result in noisy or incomplete listings, which can interfere with proper filtering and sorting.
 - ✗ May not guarantee a categorized listing specific to 'Blankets & Throws'

Selected Action: Since the next upcoming step in the meta-plan is navigating via category links and that 'Blankets & Throws' is likely a subcategory of 'Bedding', Candidate 0 is the more reliable and aligned action. Thus, Candidate 0: 'click [82]' is the correct action.

972 C.3 CURRENT INPUTS
973974 **Current Inputs**

976 **Objective:** [TASK INSTRUCTION]
 977 **Current URL:** [CURRENT URL]
 978 **Current screenshot:** [CURRENT WEBPAGE SCREENSHOT IMAGE]
 979 **Current Observation:** [CURRENT OBSERVATION]
 980 **Open tabs:** [OPEN TABS]
 981 **Previous actions:** [PREVIOUS ACTIONS]
 982 **Previous screenshots:** [PREVIOUS SCREENSHOTS]
 983 **Meta-plan:** [META-PLAN]
 984 **Target actions:** [TARGET ACTIONS]

985
986 D DYNAMIC REPLANNING OF META-PLAN
987

988 In this section, we provide the full prompt for dynamic replanning of the meta-plan. Instead of
 989 using the static meta-plan generated from Appendix A, the meta-plan is updated at every time step
 990 using below prompt, allowing the policy model to generate actions corresponding to the updated
 991 meta-plan.

992
993 **System Prompt**

994 You are an expert web navigation planner. You will be given the following inputs:
995

- 996 • **User's objective:** This is the task you're trying to complete.
- 997 • **Current screenshot:** A screenshot of the current webpage, with each interactable
998 element assigned a unique numerical id.
- 999 • **Current Observation:** Lists the IDs of all interactable elements in the format [id]
1000 [tagType] [text content]. tagType is the type of the element, such as
1001 button, link, or textbox.
- 1002 • **Previous actions:** History of actions that you have performed. It may be helpful to
1003 track your progress.
- 1004 • **Previous screenshots:** History of previous screenshots leading up to the current
1005 state by previous actions history. It may be helpful to track your progress.
- 1006 • **Meta-plan:** A structured high-level guideline outlining the logical sequence of
1007 steps you should consistently follow to effectively solve any given task.

1008
1009 **Actions you can perform:**

- 1010 • *Page Operations:* click [id], type [id] [content], hover [id],
1011 scroll [down], etc.
- 1012 • *Tab Management:* new_tab, tab_focus [index], close_tab
- 1013 • *Navigation:* goto [url], go_back, go_forward
- 1014 • *Completion:* stop [answer]

1015
1016 **Your job is to do the following:**

- 1017 1. **Identify the Current Progress Status:** Based on the previous history and the cur-
1018 rent web page screenshot, identify the last fully completed stage in the meta-plan to
1019 assess how far the agent has progressed.
- 1020 2. **Self-Verify the Next Stage**
1021 Evaluate whether the next incomplete stage in the meta-plan satisfies the following
1022 conditions based on the current observations:
 - 1023 • **✓ Executable:** Can this stage be completed by interacting with the elements
1024 currently listed in the observation?

1026 • ✓ **Essential:** Is this stage necessary for achieving the user’s objective?
 1027
 1028 If the next stage does not satisfy these conditions:
 1029 • **Revise** the plan by removing or updating the invalid stage.
 1030 • **Adjust** the subsequent stages as needed to ensure the meta-plan still leads to
 1031 successful completion of the user’s objective.
 1032
 1033 Otherwise, if the next stage is still valid and executable, keep it and all following
 1034 stages unchanged.
 1035
 1036 **Output Format:**
 1037 Format your response as a JSON array of stage objects without additional commentary.
 1038 Example:
 1039 [
 1040 {
 1041 "stage_name": "Search",
 1042 "description": "Search for the target product
 1043 using appropriate keywords"
 1044 },
 1045 {
 1046 "stage_name": "Price Comparison",
 1047 "description": "Compare prices, potentially
 1048 using sorting or filtering functionality"
 1049 },
 1050 ...
 1051]

1052 Current Inputs

1053
 1054 **Objective:** [TASK INSTRUCTION]
 1055 **Current screenshot:** [CURRENT WEBPAGE SCREENSHOT IMAGE]
 1056 **Current Observation:** [CURRENT OBSERVATION]
 1057 **Previous actions:** [PREVIOUS ACTIONS]
 1058 **Previous screenshots:** [PREVIOUS SCREENSHOTS]
 1059 **Meta-plan:** [META-PLAN]

1061 E QUALITATIVE RESULTS

1064 E.1 ACTION GENERATION

1066 In Figure 6, we visualize the agent’s trajectory of greedy reasoning and progress-aware reasoning
 1067 during the action generation process in Sec. 3.2, respectively. Notably, although the agent has
 1068 already found the ground truth target item, greedy reasoning (Gu et al., 2024) oscillates without
 1069 completing the task. In contrast, our progress-aware reasoning successfully terminates by noticing
 1070 that the last stage of the meta-plan has already been completed.

1072 E.2 ACTION SELECTION

1074 In this section, we visualize qualitative examples of the action selection process by the conventional
 1075 action value function (Gu et al., 2024) and our progress-aware action evaluator. In Figure 7, the
 1076 conventional value function greedily assigns a higher value to the stopping action, prematurely ter-
 1077 minating the task despite the objective (i.e., navigating to the detailed view of the identified item)
 1078 not being fulfilled. In contrast, our proposed action evaluation selects the appropriate `click` action
 1079 rather than the premature stop action, based on explicitly focusing on the next remaining step of the
 meta-plan (i.e., clicking the identified item), leading to the successful task completion.

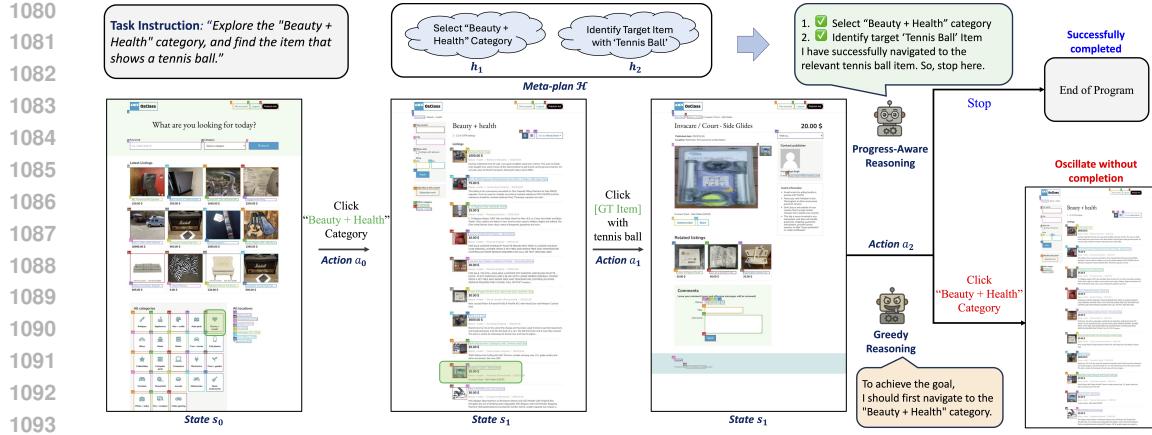


Figure 6: The trajectory of greedy reasoning (bottom; Yao et al. (2023); Gu et al. (2024); Koh et al. (2024b)) and progress-aware reasoning (top), where the agent should finalize the item search task.

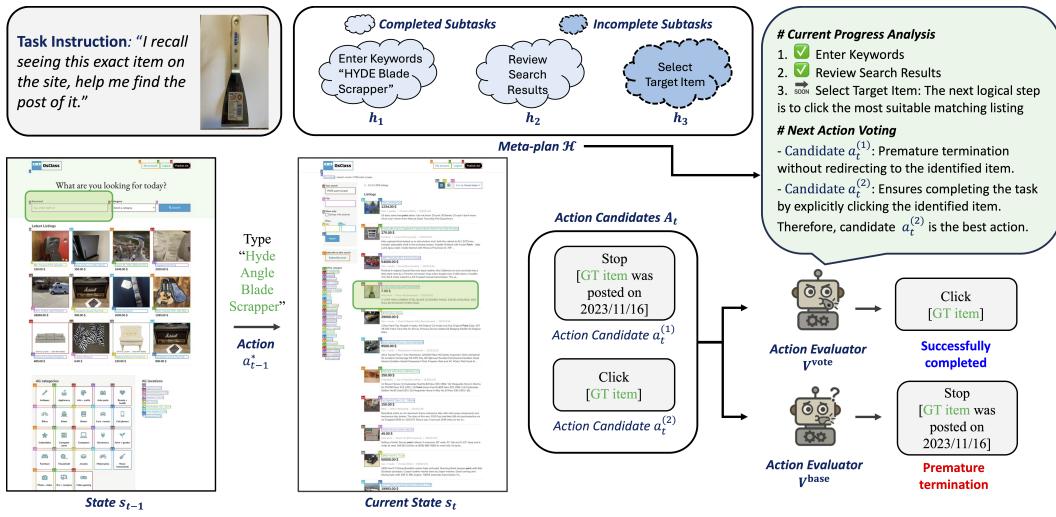


Figure 7: Action selection by conventional value function (bottom; Gu et al. (2024); Koh et al. (2024b)) and our progress-aware action evaluator (top), where the agent should finalize the item search task.

Also, in Figure 8, we note that our proposed evaluator also effectively resolves the invalid action repetition issue, where the conventional action evaluator repeatedly selects the scroll down action although the page is already reached to the bottom. However, unlike the conventional evaluator, our proposed action evaluator selects more effective action (i.e., turning to the next page) that enables further search, deviating from the invalid action loop by focusing on the next remaining subgoal in the meta-plan.

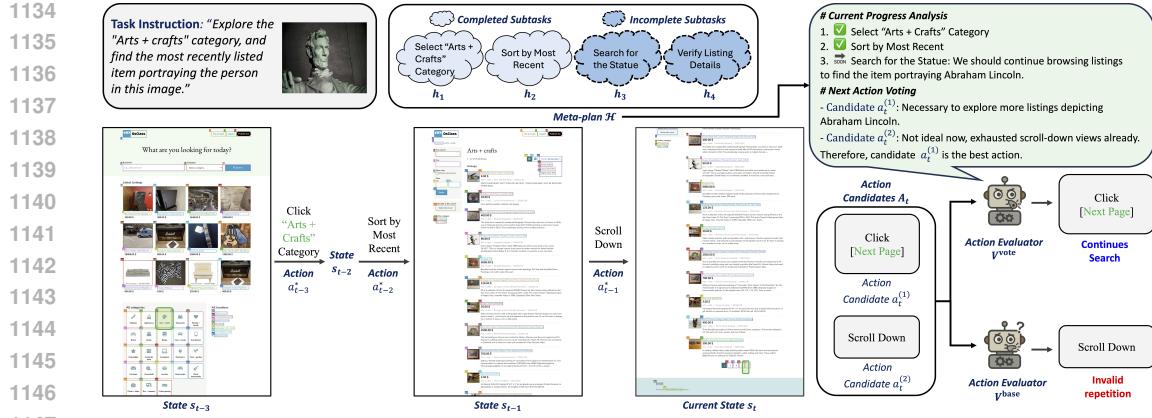


Figure 8: Action selection by conventional value function (bottom; Gu et al. (2024); Koh et al. (2024b)) and our progress-aware action evaluator (top), where the agent should continue searching for the target item on the next pages.

F RESOURCE EFFICIENCY

In Table 5, we further analyze the efficiency of MAPLE in terms of computational resource overhead. The results reveal that the previous SOTA frameworks, such as WebDreamer and TreeSearch, induce either heavy API cost or long execution time due to their additional computational components, including simulation-based rollout (Gu et al., 2024) and backtracking (Koh et al., 2024b) mechanisms. Without relying on these computationally heavy routines, our proposed method largely reduces resource overhead while achieving even higher task success rates. Specifically, MAPLE requires less than half the API cost and latency compared to WebDreamer and TreeSearch, respectively, while representing a large headroom for the success rate.

Table 5: Resource efficiency comparison with SOTA frameworks, WebDreamer and TreeSearch. We report the average of task completion time, API cost (calculated by the number of input and output tokens spent), and task success rate for each framework. For fair comparison, we used GPT-4o for all the frameworks and set the maximum number of time steps to 30.

Method	Time (sec) \downarrow	API Cost (\$) \downarrow	Classifieds SR (%) \uparrow
WebDreamer (Gu et al., 2024)	665	2.61	23.2
TreeSearch (Koh et al., 2024b)	1043	1.29	29.1
MAPLE (Eq 6, 8)	500	1.12	32.1