Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks

Zhenhailong Wang [*1] **Haiyang Xu** [*2] **Junyang Wang** [2] **Xi Zhang** [2]
Ming Yan [2] **Ji Zhang** [2] **Fei Huang** [2] **Heng Ji** [*1]

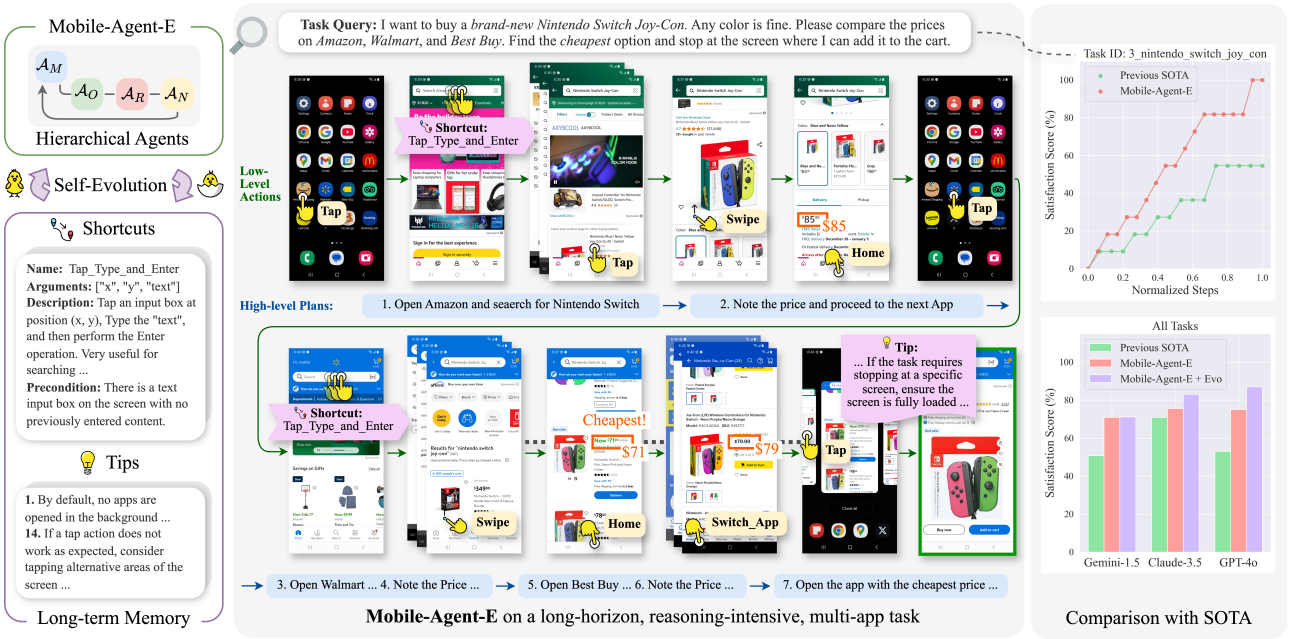


Figure 1. We propose Mobile-Agent-E, a novel hierarchical multi-agent mobile assistant that outperforms previous state-of-the-art approaches (Zhang et al., 2023; Wang et al., 2024b;a) on complex real-world tasks. Mobile-Agent-E disentangles high-level planning and low-level action decision with dedicated agents. Equipped with a newly introduced self-evolution module that learns general *Tips* and reusable *Shortcuts* from past experiences, Mobile-Agent-E demonstrates further improvements in both performance and efficiency.

Abstract

Smartphones have become indispensable in modern life, yet navigating complex, multi-step tasks on mobile devices often remains frustrating and time-consuming. Recent advancements in large multimodal model (LMM)-based mobile agents have demonstrated the ability to perceive and act in mobile environments on behalf of users. However, current approaches face significant limitations: they fall short in addressing real-world human needs, struggle with reasoning-intensive and long-horizon tasks, and lack mechanisms to learn and improve from prior experiences. To overcome these challenges, we introduce **Mobile-Agent-E**, a hierarchical multi-agent framework capable of self-evolution through past experience. By "hierarchical," we mean an explicit separation of high-level planning and low-level action execution. The framework comprises a Manager, responsible for devising overall plans by breaking down complex tasks into subgoals, and four subordinate agents—Perceptor, Operator, Action Reflector, and Notetaker—which handle fine-grained visual perception, immediate action execution, error verification, and information aggregation, respectively. Mobile-Agent-E also features a novel self-evolution module which maintains a persistent long-term memory comprising *Tips* and *Shortcuts*. Tips are general guidance and lessons learned from prior tasks on how to effectively interact with the environment. Shortcuts are reusable, executable sequences of atomic operations tailored

[1]University of Illinois Urbana-Champaign [2]Alibaba Group. [*]Corresponding authors: Zhenhailong Wang <wangz3@illinois.edu>, Haiyang Xu <shuofeng.xhy@alibaba-inc.com>, Heng Ji <hengji@illinois.edu>.

Preprint.

for specific subroutines. The inclusion of Tips and Shortcuts facilitates continuous refinement of task performance and efficiency. Alongside this framework, we introduce **Mobile-Eval-E**, a new benchmark featuring complex real-world mobile tasks requiring long-horizon, multi-app interactions. Empirical results show that Mobile-Agent-E achieves a 22% absolute improvement over previous state-of-the-art approaches across three foundation model backbones. Additionally, we provide a comprehensive analysis of the impact of our self-evolution mechanism and suggest directions for future work. Code and data are publicly available for research purposes at https://x-plug.github.io/MobileAgent.

1. Introduction

Smartphones have become integral to our daily lives, transforming the way we connect, work, and find entertainment. Yet, the average 4.5 hours people spend on their phones daily* often includes moments of frustration. Tedious tasks, such as deal hunting across multiple apps or gathering scattered information from various websites, often make us wish for a smart mobile assistant to ease these burdens. Recent advancements in large multimodal models (LMMs) (OpenAI, 2024; Anthropic, 2024; Team et al., 2024) have led to the emergence of LMM-based GUI agents (Wang et al., 2024c; Nguyen et al., 2024) capable of perceiving and acting in the Web, PC, and mobile environments on behalf of human users. Despite these initial successes, current research on mobile agents (Wang et al., 2024b; Zhang et al., 2023; Wang et al., 2024a; Li et al., 2024) has yet to fully address the challenges of real-world mobile tasks. We identify two key limitations below.

First, we observe a significant gap between the capabilities of current mobile agents and the demands of real-world scenarios. While existing mobile agent tasks are typically short, straightforward, and goal-oriented, such as "Navigate to a nearby gas station" (Wang et al., 2024a), tasks that better reflect actual human needs are far more complex. These tasks often require a combination of (1) intensive reasoning to address multiple constraints, such as balancing various factors or criteria; (2) long-horizon planning, which may involve a lengthy sequence of steps across multiple apps; and (3) exploration, where the instructions can be vague and require active information gathering rather than following a fixed trajectory. For instance, as shown in Figure 1, online shopping often involves navigating across different apps to compare prices and find the best deal. Furthermore, the highly dynamic nature of mobile environments, characterized by pop-up advertisements and frequently changing app layouts, poses additional challenges in tackling these complex real-world tasks.

Second, unlike humans, who quickly adapt and become proficient with new devices or apps, current mobile agents lack the ability to learn from prior experiences. For example, when a human user first opens an app like Maps, it may take some trial and error to understand the layout and successfully perform a search. However, with each interaction, the user learns, becoming faster and more accurate the next time. In contrast, existing mobile agents treat every task as if it were their first attempt, allocating the same computational resources at each step and repeating the same mistakes, regardless of how many times they perform the same task. This inability to accumulate knowledge and refine actions from past experiences severely limits their ability to handle the aforementioned complex, long-horizon tasks, where subroutines such as searching and creating notes are often shared across different objectives.

To address these limitations, we propose **Mobile-Agent-E**, a **hierarchical multi-agent framework** capable of **self-evolution** through past experiences. Mobile-Agent-E explicitly disentangles high-level planning—such as decomposing a task into smaller subgoals—from low-level actions, which involves determining specific actions and their parameters (e.g., `tap(x,y)`). The framework is structured with a Manager, responsible for creating overall plans, and four subordinate agents—Perceptor, Operator, Action Reflector, and Notetaker—that handle fine-grained visual perception, action decision, outcome verification, and information aggregation, respectively. This hierarchical design significantly enhances long-term planning and improves error recovery in complex tasks. Figure 1 shows an overview of Mobile-Agent-E on a challenging online shopping task requiring multi-step reasoning and interaction across three different apps.

Mobile-Agent-E also features a self-evolution module, which includes a persistent long-term memory and two Experience Reflectors. We define two types of critical knowledge that are continuously updated in the long-term memory across tasks: **Tips**—general guidance on effective interactions and lessons learned from previous trail-and-error experiences—and **Shortcuts**—reusable, executable functions that contains sequences of atomic operations tailored to efficiently complete recurring subroutines under specific preconditions. After completing each task, the Experience Reflectors are triggered to update the Tips and propose new Shortcuts based on the interaction history. These are then fed to the Manager and Operator, enabling improved planning and action decision-making in future tasks. This design draws inspiration from human cognitive science, where Tips are akin to the lessons encoded in episodic memory (Tul-

*https://explodingtopics.com/blog/smartphone-usage-stats

ving, 2002), which involves recalling specific past experiences and using them to inform future decisions, while Shortcuts resemble procedural knowledge that facilitates the efficient and often subconscious execution of well-practiced tasks (Squire & Zola, 1996; Anderson, 1982). An example of Shortcuts and Tips is provided in Figure 1.

To address the limitation of existing mobile benchmarks, which mainly include short-horizon and straightforward tasks with already saturated performance, we introduce a new benchmark, **Mobile-Eval-E**, designed to evaluate complex, real-world tasks. Mobile-Eval-E features more than *twice* the number of expected operations per task compared to previous benchmarks (Wang et al., 2024b; Zhang et al., 2023; Wang et al., 2024a) and incorporating a significantly higher proportion of tasks requiring *multi-app* interactions. Accompanying the benchmark, we introduce a new evaluation metric called the Satisfaction Score to address the challenge posed by real-world tasks that often lack a binary success flag or a ground truth trajectory. This metric is computed based on human-written rubrics that account for both milestone completion, such as "opened Maps," and exploratory behaviors, such as "viewed more than one review." This approach offers a reliable measure of agent performance aligned with human preferences. We further propose a Satisfaction Score vs Steps (SSS) curve to better evaluate and visualize the efficiency of mobile agents. Mobile-Eval-E sets a high standard of difficulty, with prior state-of-the-art methods achieving only about 50–70% of human satisfaction.

Empirical results show that Mobile-Agent-E achieves an average absolute gain of 22.1% over previous state-of-the-art approaches across three different foundation model backbones. Mobile-Agent-E also demonstrates promising self-evolution behavior in both performance and efficiency, resulting in a 6.5% absolute improvement compared to no evolution. The incorporation of Shortcuts further reduces the computational overhead, achieving speeds comparable to prior models while delivering significantly better performance. Additionally, we provide a comprehensive analysis of various aspects of self-evolution's impact and outline directions for future work.

2. Mobile-Agent-E

Figure 2 provides an overview of Mobile-Agent-E. A summary of the notation definitions is presented in Table 1. We detail the hierarchical multi-agent framework (§2.1) and the self-evolution module (§2.2) in Mobile-Agent-E below.

[†] a_t can represent either a single atomic operation or a sequence of atomic operations if performing a Shortcut.

Table 1. Notation definitions.

Notation	Description
Environment	
I	Input task query
a^t	Action[†] at time t
s^t	Phone state (screenshot) at time t
Agents	
$\mathcal{A}_P$	Perceptor
$\mathcal{A}_M$	Manager
$\mathcal{A}_O$	Operator
$\mathcal{A}_R$	Action Reflector
$\mathcal{A}_N$	Notetaker
$\mathcal{A}_{ES}$	Experience Reflector for Shortcuts
$\mathcal{A}_{ET}$	Experience Reflector for Tips
Working Memory	
W_V^t	Visual perception result at time t
W_P^t	Overall plan (decomposed subgoals) at time t
W_S^t	Current subgoal at time t
W_G^t	Progress status at time t
W_N^t	Important notes at time t
W_{EF}^t	Error Escalation Flag at time t
$\mathbf{W_A}$	Action history with outcome status
$\mathbf{W_E}$	Error history with feedback
Long-term Memory	
L_S	Shortcuts
L_T	Tips

2.1. Hierachical Multi-Agent Framework

Figure 3 provides a detailed breakdown of the main agent loop with concrete examples. Except for the Perceptor, all reasoning agents are instantiated from a frozen large multimodal model (LMM), such as GPT-4o (OpenAI, 2024). The inputs and outputs of each agent are detailed as follows.

Manager ($\mathcal{A}_M$): High-level planning. The Manager focuses on devising high-level plans to achieve the user's requests. At each step, the Manager checks the input query I, the current screenshot s_t, the previous overall plan W_P^{t-1}, the previous subgoal W_S^{t-1}, the progress status W_G^{t-1}, available Shortcuts from long-term memory L_S, and any recorded important notes W_N^{t-1} to provide an updated overall plan W_P^t and identify the next immediate subgoal W_S^t to achieve. Note that the Manager does not condition on the fine-grained perception results from the Perceptor, as it is not necessary and can add noise to high-level planning.

$$W_P^t, W_S^t = \mathcal{A}_M(I, s_t, W_P^{t-1}, W_S^{t-1}, W_G^{t-1}, W_N^{t-1}, L_S) \quad \text{if } t \geq 0 \text{ and } W_{EF}^{t-1} == \text{False} \tag{1}$$

$$W_P^t, W_S^t = \mathcal{A}_M(I, s_t, W_P^{t-1}, W_S^{t-1}, W_G^{t-1}, W_N^{t-1}, L_S, \mathbf{W_E}[-k:]) \quad \text{if } t \geq k \text{ and } W_{EF}^{t-1} == \text{True} \tag{2}$$

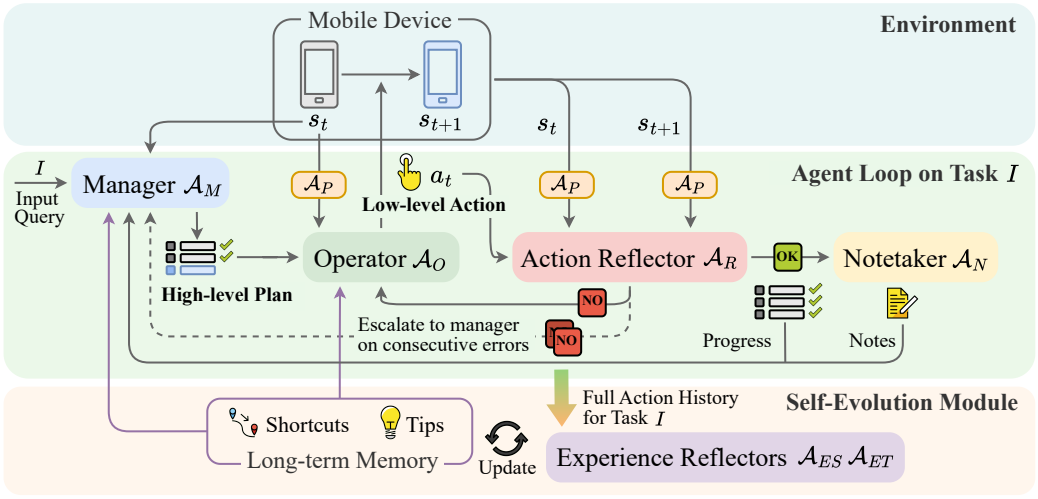


Figure 2. An overview of the Mobile-Agent-E framework, where the Manager, Perceptor ($\mathcal{A}_P$), Operator, Action Reflector, and Notetaker are involved in the main agent loop for each task, while two Experience Reflectors contribute to updating long-term memory across tasks. Decision-making at each step is disentangled into high-level planning by the Manager and low-level actions by the Operator. The Action Reflector verifies the outcome of each action, tracks progress, and provides error feedback. The Notetaker aggregates important information during navigation. A detailed example illustrating one step in the agent loop and the self-evolution process is presented in Figures 3 and 4.

Additionally, when the model is potentially stuck in an error loop, that is, observing k consecutive failed actions (e.g., $k = 2$) reported by the Action Reflector, a special Error Escalation Flag W_{EF}^{t-1} will be raised to the Manager. In such cases, the Manager will be prompted with additional information about the recent errors $\mathbf{W_E}[-k:]$ and asked to determine how to address the error from a higher-level perspective—such as refining the overall plan or adjusting the current subgoal to rectify the issue. In other cases, when an error first occurs, the Operator will attempt to address it before escalating the issue to the Manager. A concrete example of how the error escalation can help recovering from errors can be found in Figure 9.

Perceptor ($\mathcal{A}_P$): Fine-grained visual perception. The Perceptor aims to detect and recognize rich information about the current phone state, such as icons and text. We use a purely vision-based perception module that does not rely on the underlying XML file, following (Wang et al., 2024a). The Perceptor consists of three main components: an OCR model, an icon grounding model, and an icon captioning model. Given a screenshot s_t at time t, the Perceptor generates a fine-grained list of texts and icons, along with their corresponding coordinates W_V^t. Note that we still provide the original screenshot image to subsequent reasoning agents as a holistic visual context.

$$W_V^t = \mathcal{A}_P(s_t) \tag{3}$$

Operator ($\mathcal{A}_O$): Low-level action decisions. The Operator decides which concrete action to perform based on the input query I, the overall plan W_P^t and current subgoal W_S^t from the Manager, the previous progress status W_G^{t-1}, the important notes W_N^{t-1}, along with a history of the latest m actions $\mathbf{W_A}[-m:]$ and errors $\mathbf{W_E}[-m:]$.[‡] The action history includes both the action and its outcome (success or failure). The Operator is explicitly prompted to rectify errors if it observes unresolved failures in the history. The Operator also considers the *Tips* as guidance from the long-term memory, which can be self-evolved from past experiences. To enable accurate generation of the action parameters, e.g., the (x,y) coordinates on the screen for tapping, we also provide the Operator with the fine-grained perception results W_V^t from the Perceptor along with the screenshot s_t.

$$a_t = \mathcal{A}_O(I, s_t, W_V^t, W_P^t, W_S^t, W_G^t, W_N^t, \mathbf{W_A}[-m:], \mathbf{W_E}[-m:], L_S, L_T) \tag{4}$$

The output of the Operator is the next action a_t to perform. The action space is defined to contain not only *Atomic Operations* but also *Shortcuts*, which can evolve through tasks. The atomic operations include `Open_App`, `Tap`, `Swipe`, `Type`, `Enter`, `Switch_App`, `Back`, `Home`, and `Wait`. The full descriptions of the atomic operations can be found in Table 8. We detail the definitions and examples of *Shortcuts* and *Tips* in §2.2.

[‡]We empirically set $m = 5$ in our experiments.

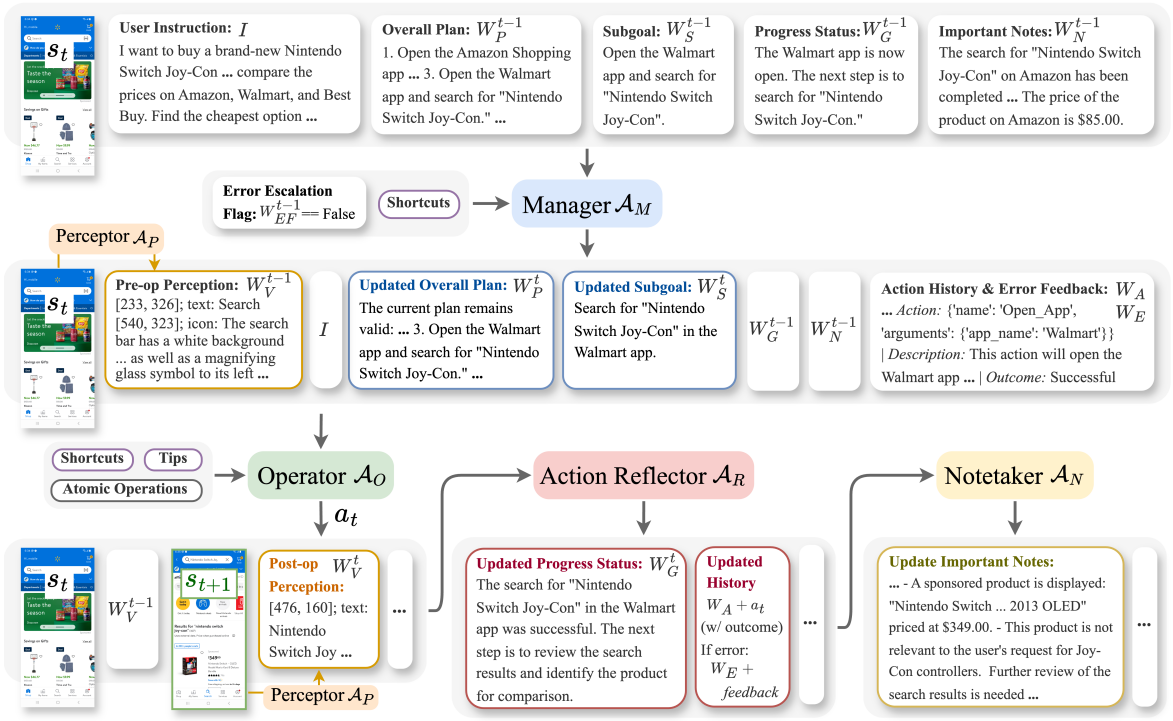


Figure 3. A detailed breakdown of one inference step t with Mobile-Agent-E, showing the inputs and outputs of each agent. Omitted information indicates no change.

Action Reflector ($\mathcal{A}_R$): Reflection on the action outcome. The Action Reflector checks the screenshots before (s_t) and after (s_{t+1}) of an action (a_t) to verify if the previous action achieves the expected outcome. We define three types of outcomes for an action: **A.** Successful or partially successful: the result of the last action meets the expectation;[§] **B.** Failed: the last action results in a wrong page; and **C.** Failed: the last action produces no changes. After identifying the outcome, if the outcome is A, the Action Reflector updates the action history $\mathbf{W_A}[t]$ as well as the progress status W_G^t. If the outcome is B or C, the Action Reflector additionally provides a description of the error and suggests potential reasons and solutions in $\mathbf{W_E}[t]$.

$$W_V^{t+1} = \mathcal{A}_P(s_{t+1}) \quad \text{\# run Perceptor on } s_{t+1} \tag{5}$$

$$\mathbf{W_A}[t], \mathbf{W_E}[t], W_G^t = \mathcal{A}_R(I, s_t, W_V^t, s_{t+1}, W_V^{t+1}, a_t, W_S^t, W_G^{t-1},) \tag{6}$$

Notetaker ($\mathcal{A}_N$): Information aggregation. In complex mobile tasks, we often need to keep track of important notes during exploration, such as the price of a product or the phone number of a restaurant. The Notetaker is dedicated to extracting and aggregating task-relevant information W_N^t after each step, based on the input query I, overall plan W_P^t, current subgoal W_S^t, current progress W_G^t, fine-grained screen perception W_V^{t+1} after executing the action, and existing notes W_N^{t-1}.

$$W_N^t = \mathcal{A}_N(I, s_{t+1}, W_V^{t+1}, W_P^t, W_S^t, W_G^t, W_N^{t-1}) \tag{7}$$

[§]Some actions may need multiple repetitions to fulfill the expectation, for example, swipe up to find reviews. Thus, we include partially successful as meeting the expectation.

2.2. Self-Evolution Module

Inspired by how humans become increasingly effective and efficient in operating smartphones, we maintain a **long-term memory** that **persists across tasks** and leverage two dedicated agents to reflect on past experiences. The long-term memory contains two important types of knowledge to evolve upon, **Tips** and **Shortcuts**, aiming to improve both the **performance** and **efficiency** of the agent. Figure 4 provides a detailed breakdown of one self-evolution step.

Tips (L_T) are defined as general guidance on effective interactions and lessons learned from previous trial-and-error experiences. Tips resemble episodic memory (Tulving, 2002), which enables humans to recall past experiences and apply insights to future decisions.

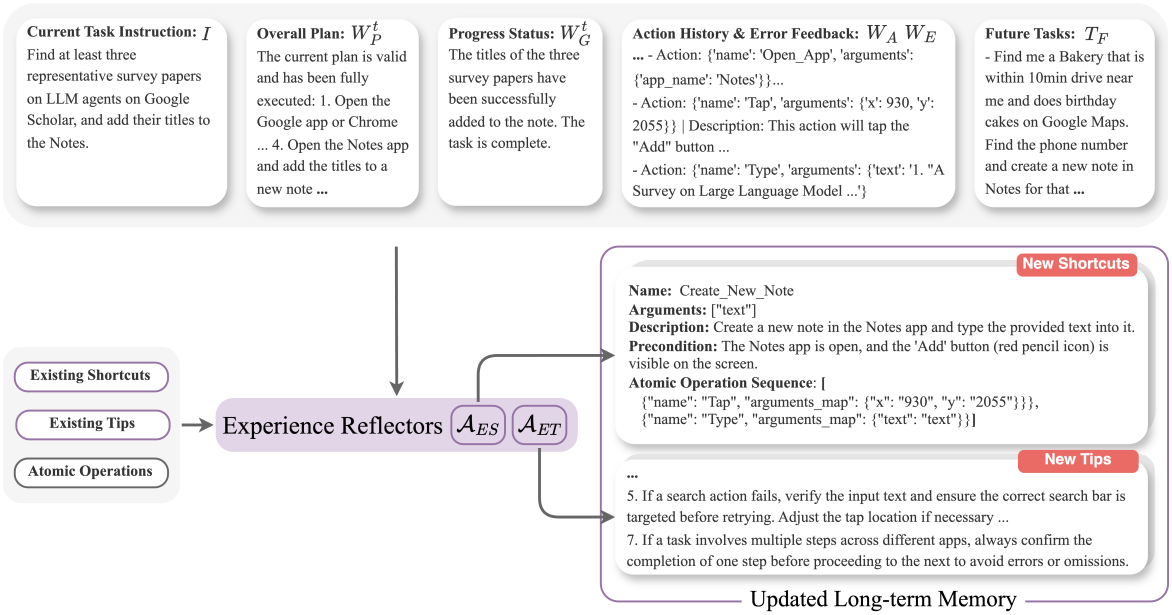


Figure 4. Illustration of the inputs and outputs to the Experience Reflectors for a single self-evolution step, including a concrete example of the newly generated Shortcuts and Tips.

Shortcuts (L_S) are defined as reusable, executable functions composed of sequences of atomic operations tailored for recurring subroutines. Shortcuts are akin to procedural knowledge, which allows humans to perform well-practiced tasks efficiently and often subconsciously (Squire & Zola, 1996; Anderson, 1982). Due to the highly dynamic nature of the mobile environment, a Shortcut may only be applicable in certain states. For instance, the "`Tap_Type_and_Enter`" Shortcut is usable only when the current screen has a text input box. To address this, we explicitly include a **precondition** in the definition of a Shortcut and require the Operator to verify that the current state satisfies the precondition before using the Shortcut. The arguments of a Shortcut have a unique one-to-one mapping to the arguments of its atomic operations.

When the self-evolution module is enabled, we leverage two Experience Reflectors, $\mathcal{A}_{ES}$ and $\mathcal{A}_{ET}$, to update the Tips and Shortcuts at the end of each task. The Experience Reflectors are also instantiated from frozen large multimodal model such as GPT-4o. Let the final time step of a task be $t = \tau$. The input to the Experience Reflectors includes the input query I, the final overall plan W_P^τ, the final progress status W_G^τ, the entire action history $\mathbf{W_A}$ and error history $\mathbf{W_E}$, the existing Shortcuts L_S and Tips L_T, and a list of future tasks T_F (if provided). The outputs consist of newly generated Shortcuts in a predefined JSON format and updated Tips in natural language. Figures 12 and 13 shows a full list of generated Shortcuts and Tips by Mobile-Agent-E.

$$L_T = \mathcal{A}_{ET}(I, W_P^\tau, W_G^\tau, \mathbf{W_A}, \mathbf{W_E}, T_F, L_T) \quad (8)$$

$$L_S = \mathcal{A}_{ES}(I, W_P^\tau, W_G^\tau, \mathbf{W_A}, \mathbf{W_E}, T_F, L_S) \quad (9)$$

The updated Tips and Shortcuts are then utilized by the Manager and the Operator in the subsequent task, facilitating evolution in both high-level planning and low-level action decisions.

3. Experiments

We perform a dynamic evaluation, evaluating the models in real-time and on actual devices—following previous work (Wang et al., 2024b; Zhang et al., 2023; Wang et al., 2024a). Specifically, we use the Android Debug Bridge (ADB) to control an Android phone[¶] and perform human evaluation on the recorded screenshots and action histories.

3.1. A More Challenging Benchmark: Mobile-Eval-E

Existing dynamic benchmarks (Wang et al., 2024b; Zhang et al., 2023; Wang et al., 2024a) primarily focus on short-horizon, straightforward tasks, where the performance has already saturated. To address this limitation, we propose a new and challenging benchmark, **Mobile-Eval-E**, which emphasizes reasoning-intensive, long-horizon, multi-app tasks. Mobile-Eval-E comprises 25 manually crafted tasks spanning 5 real-world scenarios: "Restaurant Recom-

[¶]A Samsung Galaxy A15 is used for all experiments.

Table 2. Comparison with existing dynamic evaluation benchmarks on real devices. Mobile-Eval-E emphasizes long-horizon, complex tasks that require significantly more operations and a wider variety of apps.

Benchmark	#Tasks	#Multi-App Tasks	#Apps	Avg # Ops	Total # Ops
Mobile-Eval	33	3	10	5.55	183
Mobile-Eval-v2	44	4	10	5.57	245
AppAgent	**45**	0	9	6.31	284
Mobile-Eval-E	25	**19**	**15**	**14.56**	**364**

mendation", "Information Searching", "Online Shopping", "What's Trending", and "Travel Planning". As shown in Table 2, Mobile-Eval-E significantly surpasses previous benchmarks in complexity, featuring more than $\mathbf{2}\times$ the number of expected operations per task and a greater total number of operations. Most tasks in existing benchmarks can be viewed as specific subgoals in Mobile-Eval-E. Additionally, Mobile-Eval-E encompasses a broader range of Apps, with 76% of the tasks requiring interactions with multiple Apps—compared to less than 10% in previous benchmarks. In §3, we demonstrate that this benchmark presents a substantial challenge for existing state-of-the-art models. The full set of task queries can be found in Appendix Table 7. Due to the long-horizon nature of the tasks, we keep the number of tasks relatively small to ensure a reasonable human evaluation workload for fine-grained analysis.

3.2. Metrics with Better Human Alignment

Previous dynamic evaluation typically employs a binary success rate or a completion rate against a "ground truth" trajectory to evaluate the level of task completeness. However, real-world tasks often do not have a binary success flag or a single ground truth action sequence. For example, some tasks, such as "Plan a one-day itinerary for Palo Alto," may involve exploration and information aggregation, where multiple reasonable solutions might exist. Thus, we seek to measure *human satisfaction* rather than exact matches with a ground truth trajectory. For each task, we first manually write a list of rubrics (an example shown in Figure 5(a)), containing both milestone steps (e.g., "Opened Tripadvisor") and satisfaction criteria (e.g., "Viewed multiple attractions"). We then define the **Satisfaction Score (SS)** as the number of fulfilled rubrics divided by the total number of rubrics, as judged by a human evaluator.

We also include **Action Accuracy (AA)** and **Reflection Accuracy (RA)** as metrics to evaluate action-level performance. These metrics are also assessed by humans through a review of recorded screenshots and action histories. Finally, we include a **Termination Error (TE)** rate to reflect the agent's robustness and error recovery capability. There are five ways an agent can exit from performing a task: (1) self-reported success: the agent decides to stop on its own; (2) reaching the maximum number of iterations: we set the maximum iteration count to 40 to prevent infinite loops; (3) reaching the maximum number of consecutive errors: if the agent has an action reflector and it identifies 3 consecutive errors, the agent is exited; (4) reaching the maximum number of repeated actions: if the agent performs the exact same action (excluding `Swipe` and `Back`) more than 3 consecutive times; (5) any other errors, such as errors when parsing the raw response into a valid action. If a task exits in one of the ways described in 2–5, it is marked as having a termination error. The TE rate is computed as the ratio of tasks with termination errors to all tasks.

3.3. Evaluating Self-Evolving Mobile Agents

To the best of our knowledge, this is the first work exploring evaluation in cross-task evolution settings. We consider two variants of Mobile-Agent-E: with and without the self-evolution module. When self-evolution module is enabled—referred to as *Mobile-Agent-E + Evo*—the agent performs sequentially across tasks within each scenario from the Mobile-Eval-E benchmark. The five tasks in a scenario share a persistent long-term memory. At the end of the k-th task, the Experience Reflectors are prompted to update the long-term memory based on the interaction history of the current task as well as the queries for the remaining $5-k$ tasks. This mimics the implicit requirement for an evolving agent to plan ahead, storing relevant knowledge for future interactions. In this setting, tasks performed later in the sequence benefit from a greater accumulation of Tips and Shortcuts, enabling us to analyze the progressive impact of self-evolution over time (detailed in Figure 6).

3.4. Models

Baselines. We compare against a wide range of open-sourced mobile agent frameworks, including AppAgent (Zhang et al., 2023), Mobile-Agent-v1 (Wang et al., 2024b), and Mobile-Agent-v2 (Wang et al., 2024a). To maximize an apple-to-apple comparison with Mobile-Agent-v2, which is the previous state-of-the-art, we apply an identical atomic operation space, perception model, and initial Tips to Mobile-Agent-v2 as Mobile-Agent-E. AppAgent originally requires an additional exploration phase, which does not fit our setting; thus, we add the initial Tips as additional knowledge.

Backbones. We explore using various large multimodal models (LMM) as backbones for the reasoning agents, including GPT-4o (OpenAI, 2024)[‖], Claude-3.5-Sonnet (An-

[‖]GPT-4o version: gpt-4o-2024-11-20

Table 3. Comparison with state-of-the-art models on the Mobile-Eval-E benchmark, using GPT-4o as the backbone. Mobile-Agent-E outperforms previous SOTA models by a significant margin across all metrics, demonstrating superior long-term planning, decision accuracy, and error recovery. Enabling self-evolution (*Mobile-Agent-E + Evo*) further enhances performance. Reflection Accuracy for AppAgent and Mobile-Agent-v1 are omitted since they do not have action reflectors.

Model	Type	Satisfaction Score (%) ↑	Action Accuracy (%) ↑	Reflection Accuracy (%) ↑	Termination Error (%) ↓
AppAgent (Zhang et al., 2023)	Single-Agent	25.2	60.7	-	96.0
Mobile-Agent-v1 (Wang et al., 2024b)	Single-Agent	45.5	69.8	-	68.0
Mobile-Agent-v2 (Wang et al., 2024a)	Multi-Agent	53.0	73.2	96.7	52.0
Mobile-Agent-E	Multi-Agent	75.1	85.9	97.4	32.0
Mobile-Agent-E + Evo	Multi-Agent	**86.9**	**90.4**	**97.8**	**12.0**

Table 4. Results on different large multimodal model backbones, including GPT-4o, Gemini, and Claude. The metrics *SS*, *AA*, *RA*, and *TE* represent Satisfaction Score, Action Accuracy, Reflection Accuracy, and Termination Error, respectively, expressed as percentages.

Model	Gemini-1.5-pro				Claude-3.5-Sonnet				GPT-4o			
	SS↑	AA↑	RA↑	TE↓	SS↑	AA↑	RA↑	TE↓	SS↑	AA↑	RA↑	TE↓
Mobile-Agent-v2 (Wang et al., 2024a)	50.8	63.4	83.9	64.0	70.9	76.4	96.9	32.0	53.0	73.2	96.7	52.0
Mobile-Agent-E	70.9	74.3	**91.3**	**48.0**	75.5	91.1	99.1	**12.0**	75.1	85.9	97.4	32.0
Mobile-Agent-E + Evo	**71.2**	**77.4**	89.6	**48.0**	**83.0**	**91.4**	**99.7**	**12.0**	**86.9**	**90.4**	**97.8**	**12.0**

thropic, 2024)[**], and Gemini-1.5-pro (Team et al., 2024)[††]. Unless otherwise specified, the default backbone for all models is GPT-4o.

Perceptor Implementation in Mobile-Agent-E. We closely follow Mobile-Agent-v2 (Wang et al., 2024a) to implement the Perceptor with slight modifications. We use DBNet[‡‡](Liao et al., 2020) and ConvNextViT-document[§§] from ModelScope for OCR detection and recognition respectively. We use GroundingDINO (Liu et al., 2023) for icon grounding and Qwen-VL-Plus (Bai et al., 2023) for generating captions for each cropped icon.

4. Results

4.1. Evaluation on Performance

Comparison with state-of-the-art. Table 3 presents the results on Mobile-Eval-E using an identical GPT-4o backbone for all baselines and Mobile-Agent-E. Mobile-Agent-E outperforms the previous multi-agent state-of-the-art (SOTA) model (Wang et al., 2024a) by **22.1%** in the Satisfaction Score. This comparison particularly highlights the effectiveness of the hierarchy in our multi-agent framework. Our approach also demonstrates superior robustness and error recovery capabilities, as indicated by a significantly lower termination error rate. Moreover, enabling self-evolution further enhances performance, leading to an improvement of **33.9%** against the previous SOTA, underscoring the benefit of learning from experience. In §4.3, we provide further analysis of the impact of the evolution module.

Varying reasoning backbones. Table 4 shows the comparison with previous SOTA (Wang et al., 2024a) using various backbone LMMs. We observe consistent improvements on all recent LMMs, including GPT-4o, Claude-3.5-Sonnet, and Gemini-1.5-pro, with average absolute gains of **22.1%** and **15.6%** with and without evolution, respectively. Additionally, the benefits of self-evolution appear to be more pronounced in stronger backbones, such as GPT-4o and Claude.

4.2. Evaluation on Efficiency

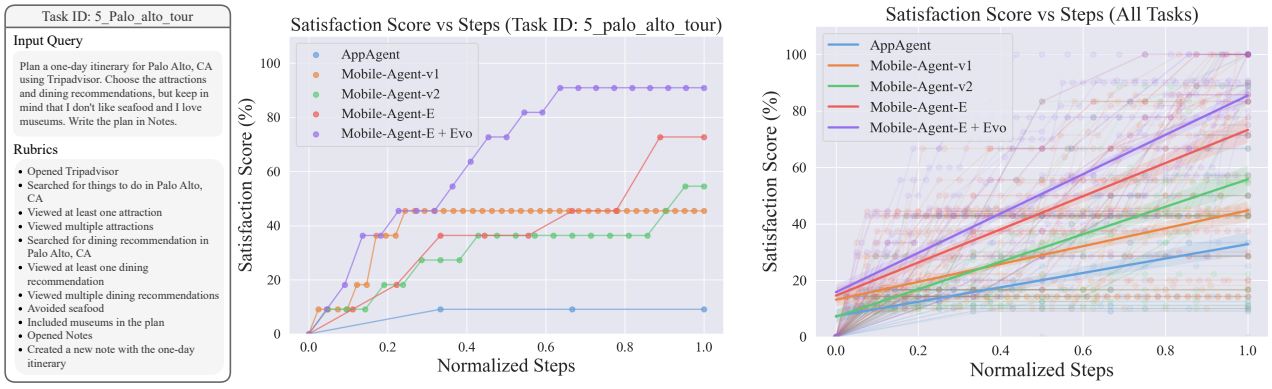
Evaluating the efficiency of mobile agents on complex, potentially open-ended tasks is not straightforward. Merely counting the number of steps is not optimal, as many tasks require exploration. A smaller number of steps reflects a quick exit but may result in insufficient exploration. Intuitively, if an agent achieves higher satisfaction, i.e., fulfills more rubrics, in a smaller number of steps, it is considered more efficient. Thus, we introduce the **Satisfaction Score vs Steps (SSS) curve** to compare and visualize the efficiency of different agents. To plot the SSS curve, we manually examine the recorded trajectories and track the satisfaction of rubrics after each step. We then plot a poly-line with the step number as the x-axis and the Satisfaction Score as the y-axis. To enable visualization of trajectories with different

[**]Claude-3.5 version: claude-3-5-sonnet-20241022

[††]Gemini-1.5 version: gemini-1.5-pro-latest (Dec 2024)

[‡‡]https://modelscope.cn/models/iic/cv_resnet18_ocr-detection-db-line-level_damo

[§§]https://modelscope.cn/models/iic/cv_convnextTiny_ocr-recognition-document_damo



(a) Single Task (task id: 5_palo_alto_tour) (b) All Tasks

Figure 5. Satisfaction Score vs. Steps (SSS) curve for (a) a single task and (b) all tasks. In (a), we also provide a concrete example of the human-written rubrics for the task, which are used to compute the Satisfaction Score during human evaluation. In (b), we additionally include a linear regression line for each model; a steeper and higher line indicates better efficiency for completing the task.

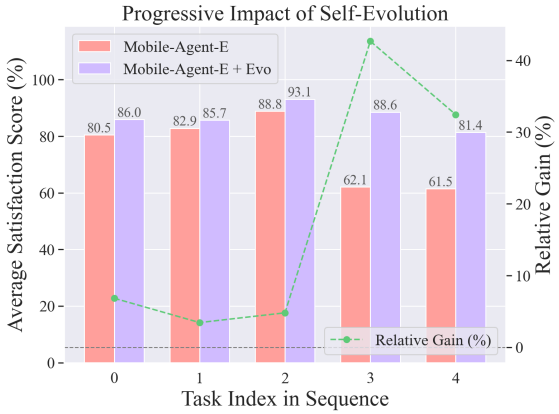


Figure 6. Progressive impact of self-evolution over time. The task index represents the order in which a task is performed in the evolution setting. The results demonstrate that tasks performed later in the sequence show more significant improvements, highlighting the increased benefits from additional iterations of self-evolution.

lengths on the same graph, we normalize the steps to the range [0, 1]. The y-axis of the rightmost point indicates the final satisfaction score. Intuitively, **a steeper and higher SSS curve indicates better efficiency and completeness**. As shown in Figure 5, we observe that Mobile-Agent-E not only achieves better final performance but also fulfills rubrics faster.

4.3. Further Analysis

Progressive impact of self-evolution over time. The ideal behavior of self-evolution is to progressively bring more benefits to the agent as knowledge accumulates. To investigate this, we group the results of the tasks by their ordering index in each scenario and compare the performance with and without enabling the evolution module. In Figure 6, the x-axis reflects the task index in the sequence

Table 5. Analysis of computational overhead and Shortcut usage. In the inference speed table, the *reasoning only* section accounts for time spent solely on reasoning agents, while *perception + reasoning* includes the runtime of the Perceptor **on CPU**. Shortcut usage statistics are calculated as the ratio of Shortcuts used to the total number of actions performed by the Operator. The use of Shortcuts significantly accelerates inference, achieving comparable times to previous, simpler frameworks.

Inference Speed (Seconds per operation)						
Model	*Reasoning Only*			*Perception + Reasoning*		
	Gemini	Claude	GPT	Gemini	Claude	GPT
Mobile-Agent-v2	9.8	21.4	12.3	25.6	38.4	43.5
Mobile-Agent-E	16.5	25.5	17.4	30.8	41.0	30.1
Mobile-Agent-E + Evo	12.9	24.8	14.9	27.2	39.6	27.4

Shortcut Usage Percentage (%)			
Model	Gemini	Claude	GPT
Mobile-Agent-E	11.9	12.8	12.4
Mobile-Agent-E + Evo	14.8	13.2	14.4

it is performed, with later tasks having access to Tips and Shortcuts that are updated through more tasks. We observe a generally increasing trend indicating that the gain tends to be more significant in later tasks, demonstrating that the self-evolution module is capable of continuously improving the agent as it experiences more tasks. The gain is not strictly monotonically increasing, as expected, since the difficulty of tasks at different indices varies.

Shortcut reduces computational overhead. The hierarchical multi-agent architecture in Mobile-Agent-E significantly improves performance on complex tasks but inevitably increases computational complexity. However, we found that the use of Shortcuts largely mitigates this overhead, enabling Mobile-Agent-E to achieve a speed comparable to that of previous models. In Table 5, we report the

Table 6. To investigate the unique impact of Tips, we compute the Satisfaction Score on a subset of instances where no newly generated Shortcuts are used in the trajectory. The results show distinctive benefits from the evolved Tips.

	Gemini	Claude	GPT-4o
Mobile-Agent-E	69.0	75.6	79.7
Mobile-Agent-E + evolved Tips	**72.6**	**85.2**	**87.5**

seconds per operation averaged across all tasks as well as the usage of Shortcuts. We observe a positive correlation between using more Shortcuts and faster inference speed. This is because a Shortcut enables the execution of multiple operations within a single decision-making iteration. For example, using the `Tap_Type_and_Enter` Shortcut to perform a search subroutine saves two iterations of perception and reasoning compared to using three atomic actions: `Tap`, `Type`, and `Enter`.

Unique impact from Tips. While the impact from Shortcuts is directly visible in the action history, it is less obvious whether the evolved Tips bring distinctive benefits. To visualize this, we filter out task instances where the same set of unique Shortcuts is used or where only atomic actions are employed, and compare the Satisfaction Score with or without the evolution. Table 6 shows that Tips alone serve as an important aspect of self-evolution.

4.4. Case Study: A Closed-Loop Self-Evolving Agent

In real-world mobile usage, after running the agent on a large number of tasks in various scenarios, the accumulated Tips and Shortcuts may grow to an amount where it is no longer feasible to include all of them in the decision-making context. Thus, in this case study, we aim to explore closing the self-evolution loop by introducing two additional **Experience Retriever** agents for Tips $\mathcal{A}_{ERT}$ and Shortcuts $\mathcal{A}_{ERS}$. We consider a new task in an unknown scenario, as shown in Figure 7. First, we provide all the updated Tips and Shortcuts—after running Mobile-Agent-E on all 5 scenarios (a total of 25 tasks) in Mobile-Eval-E—to the Experience Retrievers. With GPT-4o as the backbone, the updated long-term memory contains a total of 7 unique Shortcuts and 58 Tips, among which 6 Shortcuts and 55 Tips are newly proposed by Mobile-Agent-E during experience reflection. Then, the Experience Retrievers are prompted to select only the relevant Tips and Shortcuts for the current task. The qualitative example in Figure 7 shows that Mobile-Agent-E effectively retrieves and leverages highly relevant Shortcuts and Tips to successfully complete a challenging unseen task. The full list of Tips and Shortcuts after evolution can be found in Appendices G and F.

5. Related Work

5.1. GUI Agents

The advancement of large multimodal models (LMM) has introduced a new area of agentic research focused on LMM-based GUI agents (Wang et al., 2024c). The goal is to develop AI assistants capable of performing tasks in various GUI environments, such as Web (Deng et al., 2023; Zheng et al., 2024; He et al., 2024; Yoran et al., 2024; Reddy et al., 2024), PC (Hong et al., 2023; Zhang et al., 2024; Liu et al., 2024b; Xie et al., 2024; Tan et al., 2024), and mobile devices (Wang et al., 2024b; Zhang et al., 2023; Li et al., 2024; Wang et al., 2024a; Liu et al., 2024a). In the mobile environment, one line of research focuses on improving the perception and reasoning abilities of a single agent through tool usage (Wang et al., 2024b) and an additional exploration phase (Zhang et al., 2023; Li et al., 2024). Recent studies (Rawles et al., 2024; Wang et al., 2024a) show significant promise by incorporating multiple agents for decision-making and reflection. However, current multi-agent frameworks still face challenges such as short-sighted planning and poor error recovery. Specifically, the "planning" module in Mobile-Agent-v2 (Wang et al., 2024a) functions primarily as a progress tracker, while the "decision-making" module continues to handle both high-level planning (e.g., "what to do next") and low-level action decisions (e.g., "where to tap"). A key difference in Mobile-Agent-E is the introduction of a *hierarchy* among the agents, enabling more effective long-horizon planning and improved low-level action accuracy.

5.2. Self-Evolution in Foundation Models

Investigating how to make large language models and multimodal models self-improve has long been an active area of research (Tao et al., 2024). One line of work focuses on enhancing the base abilities of foundation models, such as improving reasoning and reducing knowledge hallucination. This includes approaches like iterative refinement (Madaan et al., 2024), self-reflection (Shinn et al., 2024), self-training (Huang et al., 2022), self-improvement (Wang et al., 2024d), and multi-persona collaboration (Wang et al., 2023). Another line of work explores improving task-solving with foundation models through tool learning and tool creation (Cai et al., 2023; Qian et al., 2023; Yuan et al., 2023). In the context of GUI agents, self-evolution is less studied. The skill curation mechanism in Cradle (Tan et al., 2024) shows initial promise in the PC environment; however, no previous work has systematically explored self-evolution in mobile environments. In this work, we demonstrate the importance of self-evolution in both Tips and Shortcuts. Notably, unlike the "skills" in Cradle, which are directly added to the atomic operation space, we explicitly define *preconditions* for our Shortcuts, as this is critical for decision-making

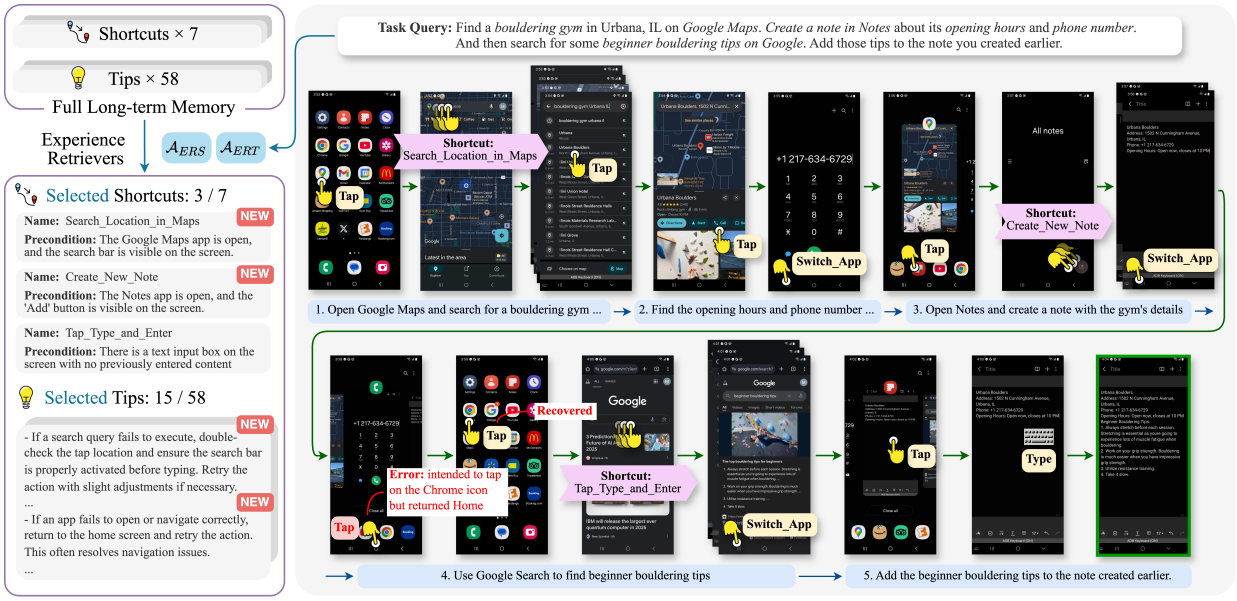


Figure 7. Case study example where relevant Shortcuts and Tips are automatically retrieved from the previously evolved long-term memory and subsequently leveraged to complete an unseen, challenging task. The action trajectory also includes an example where the agent recovers from an error.

across multiple apps and varying layouts.

6. Conclusion and Future Work

We introduce Mobile-Agent-E, a novel mobile assistant featuring a hierarchical multi-agent framework and a self-evolution module that significantly enhances long-term planning, error recovery, and efficiency, excelling in a wide variety of complex real-world tasks. Remaining limitations include the incorrect usage of Shortcuts with invalid preconditions and erroneous agent-generated Shortcuts, with detailed examples provided in Appendix C. Future work will focus on developing improved strategies for generating, invoking, and revising Shortcuts, enhancing personalization to better adapt to individual user needs, and strengthening safety precautions to enable more effective human-agent collaboration.

Impact Statement

This paper aims to advance the field of LMM-based agents by developing a hierarchical multi-agent framework and benchmark to improve the usability and efficiency of smartphones in complex, multi-step tasks. While the primary goal is to enhance human-device interaction, the proposed system has the potential for broader societal benefits, particularly in improving accessibility for individuals with disabilities or limited mobility. By enabling more intuitive and automated task management on mobile devices, this framework can assist users with physical impairments, cognitive challenges, or conditions that make precise interactions with touchscreens difficult.

While the primary aim is to enhance mobile task efficiency and user accessibility, the development of mobile agents capable of autonomous decision-making introduces potential risks. For example, unauthorized or unintended actions by the agent, such as the misuse of sensitive information including credit card details or private data, could result in serious consequences for users. These risks emphasize the critical need for robust safeguards, error recovery mechanisms, and fail-safe systems to ensure that the agent's actions consistently align with user intentions.

We are actively pursuing future work that focuses on designing and integrating robust privacy and safety mechanisms. These include explicit user consent workflows for sensitive operations, encryption protocols to protect user data during processing and storage, and automated systems to flag potentially harmful or unauthorized actions. These advancements will be crucial for maximizing the societal benefits of these systems, minimizing potential risks, and building user trust in autonomous mobile agents.

References

Anderson, J. R. Acquisition of cognitive skill. *Psychological review*, 89(4):369, 1982.

Anthropic. Claude 3.5 Sonnet, 2024. URL https://www.anthropic.com/news/3-5-models-and-computer-use.

Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., and Zhou, J. Qwen-VL: A Frontier Large Vision-Language Model with Versatile Abilities. *arXiv preprint arXiv:2308.12966*, 2023. URL https://doi.org/10.48550/arXiv.2308.12966.

Cai, T., Wang, X., Ma, T., Chen, X., and Zhou, D. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*, 2023.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=kiYqbO3wqw.

He, H., Yao, W., Ma, K., Yu, W., Dai, Y., Zhang, H., Lan, Z., and Yu, D. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., and Tang, J. Cogagent: A visual language model for gui agents, 2023.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Li, Y., Zhang, C., Yang, W., Fu, B., Cheng, P., Chen, X., Chen, L., and Wei, Y. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*, 2024.

Liao, M., Wan, Z., Yao, C., Chen, K., and Bai, X. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11474–11481, 2020.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., and Zhang, L. Grounding DINO: marrying DINO with grounded pre-training for open-set object detection. *CoRR*, abs/2303.05499, 2023. doi: 10.48550/ARXIV.2303.05499. URL https://doi.org/10.48550/arXiv.2303.05499.

Liu, X., Qin, B., Liang, D., Dong, G., Lai, H., Zhang, H., Zhao, H., Iong, I. L., Sun, J., Wang, J., et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024a.

Liu, X., Zhang, T., Gu, Y., Iong, I. L., Xu, Y., Song, X., Zhang, S., Lai, H., Liu, X., Zhao, H., et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024b.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Nguyen, D., Chen, J., Wang, Y., Wu, G., Park, N., Hu, Z., Lyu, H., Wu, J., Aponte, R., Xia, Y., et al. Gui agents: A survey. *arXiv preprint arXiv:2412.13501*, 2024.

OpenAI. GPT-4o System Card, 2024. URL https://cdn.openai.com/gpt-4o-system-card.pdf.

Qian, C., Han, C., Fung, Y. R., Qin, Y., Liu, Z., and Ji, H. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. *arXiv preprint arXiv:2305.14318*, 2023.

Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W., Li, W., Campbell-Ajala, F., et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

Reddy, R. G., Mukherjee, S., Kim, J., Wang, Z., Hakkani-Tur, D., and Ji, H. Infogent: An agent-based framework for web information aggregation. *arXiv preprint arXiv:2410.19054*, 2024.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Squire, L. R. and Zola, S. M. Structure and function of declarative and nondeclarative memory systems. *Proceedings of the National Academy of Sciences*, 93(24): 13515–13522, 1996.

Tan, W., Ding, Z., Zhang, W., Li, B., Zhou, B., Yue, J., Xia, H., Jiang, J., Zheng, L., Xu, X., et al. Towards general computer control: A multimodal agent for red dead redemption ii as a case study. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Tao, Z., Lin, T.-E., Chen, X., Li, H., Wu, Y., Li, Y., Jin, Z., Huang, F., Tao, D., and Zhou, J. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*, 2024.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S.,

et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Tulving, E. Episodic memory: From mind to brain. *Annual review of psychology*, 53(1):1–25, 2002.

Wang, J., Xu, H., Jia, H., Zhang, X., Yan, M., Shen, W., Zhang, J., Huang, F., and Sang, J. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv preprint arXiv:2406.01014*, 2024a.

Wang, J., Xu, H., Ye, J., Yan, M., Shen, W., Zhang, J., Huang, F., and Sang, J. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024b.

Wang, S., Liu, W., Chen, J., Gan, W., Zeng, X., Yu, S., Hao, X., Shao, K., Wang, Y., and Tang, R. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*, 2024c.

Wang, Z., Mao, S., Wu, W., Ge, T., Wei, F., and Ji, H. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*, 2023.

Wang, Z., Hou, L., Lu, T., Wu, Y., Li, Y., Yu, H., and Ji, H. Enable lanuguage models to implicitly learn self-improvement from data. In *Proc. The Twelfth International Conference on Learning Representations (ICLR2024)*, 2024d.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

Yoran, O., Amouyal, S. J., Malaviya, C., Bogin, B., Press, O., and Berant, J. Assistantbench: Can web agents solve realistic and time-consuming tasks?, 2024. URL https://arxiv.org/abs/2407.15711.

Yuan, L., Chen, Y., Wang, X., Fung, Y. R., Peng, H., and Ji, H. Craft: Customizing llms by creating and retrieving from specialized toolsets. *arXiv preprint arXiv:2309.17428*, 2023.

Zhang, C., Yang, Z., Liu, J., Han, Y., Chen, X., Huang, Z., Fu, B., and Yu, G. Appagent: Multimodal agents as smartphone users, 2023.

Zhang, C., Li, L., He, S., Zhang, X., Qiao, B., Qin, S., Ma, M., Kang, Y., Lin, Q., Rajmohan, S., Zhang, D., and Zhang, Q. UFO: A UI-Focused Agent for Windows OS Interaction. *arXiv preprint arXiv:2402.07939*, 2024.

Zheng, B., Gou, B., Kil, J., Sun, H., and Su, Y. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=piecKJ2DlB.

A. Full Trajectory Comparison Example with Previous SOTA

Figure 8 presents the full trajectory of the task shown in Figure 1, comparing the previous state-of-the-art, Mobile-Agent-v2 (Wang et al., 2024a), and our proposed Mobile-Agent-E. Mobile-Agent-v2 suffers from early termination after interacting with two Apps, whereas Mobile-Agent-E fulfills all rubrics and stops at the App offering the best deal.

B. Error Recovery with Escalation to Manager

Figure 9 illustrates how the error escalation mechanism in Mobile-Agent-E enhances error recovery ability. A detailed description of the example is provided in the caption.

C. Remaining Limitations

C.1. Misuse of Shortcuts due to Incorrect Perception of Phone State

Although we explicitly require the Operator to verify the current phone state to ensure it fulfills the *precondition* of a Shortcut before calling it, there are still cases where the model incorrectly perceives the state, resulting in the misuse of Shortcuts in an invalid state. Figure 10 illustrates an example of such error. A detailed description of the example is provided in the caption. This type of error could potentially be mitigated by employing a dedicated agent for verifying preconditions or by enhancing the perception module to better understand phone states.

C.2. Errors and Imperfections in Self-Evolved Shortcuts

Although effective in most cases, we still observe errors and imperfections in the agent-generated Shortcuts during self-evolution. These issues can lead to propagated errors when an erroneous Shortcut is used in subsequent tasks. Figure 11 illustrates an example of such erroneous and imperfect Shortcuts. A detailed description of the example is provided in the caption. This highlights the need for future work on approaches to generate higher-quality Shortcuts and equipping the agent with the ability to reflect on and revise generated Shortcuts in subsequent tasks.

D. All Tasks in Mobile-Eval-E Benchmark

Table 7 presents the input queries, involved App types, and scenarios for all Mobile-Eval-E tasks. The complete list of rubrics and human reference operation sequences is provided in the supplementary material.

E. Atomic Operation Space

Table 8 presents all atomic operations considered in Mobile-Agent-E.

F. Full list of Self-Evolved Shortcuts

Figure 12 shows a full list of generated Shortcuts by Mobile-Agent-E after self-evolution on all 25 tasks from Mobile-Eval-E benchmark.

G. Full list of Self-Evolved Tips

Figure 13 shows a full list of generated Tips by Mobile-Agent-E after self-evolution on all 25 tasks from Mobile-Eval-E benchmark.

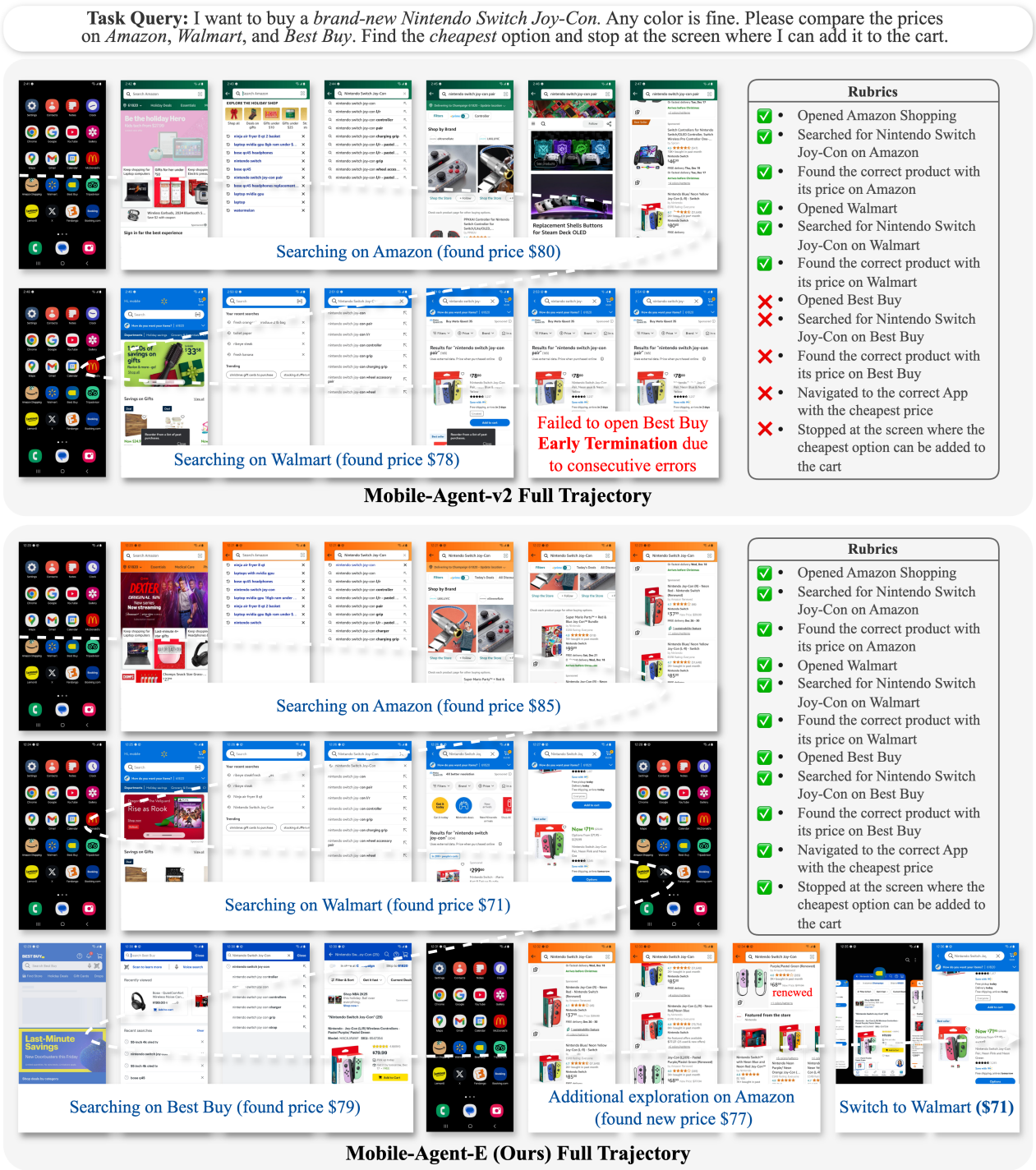


Figure 8. Full trajectory comparison between the previous state-of-the-art, Mobile-Agent-v2 (Wang et al., 2024a), and Mobile-Agent-E.

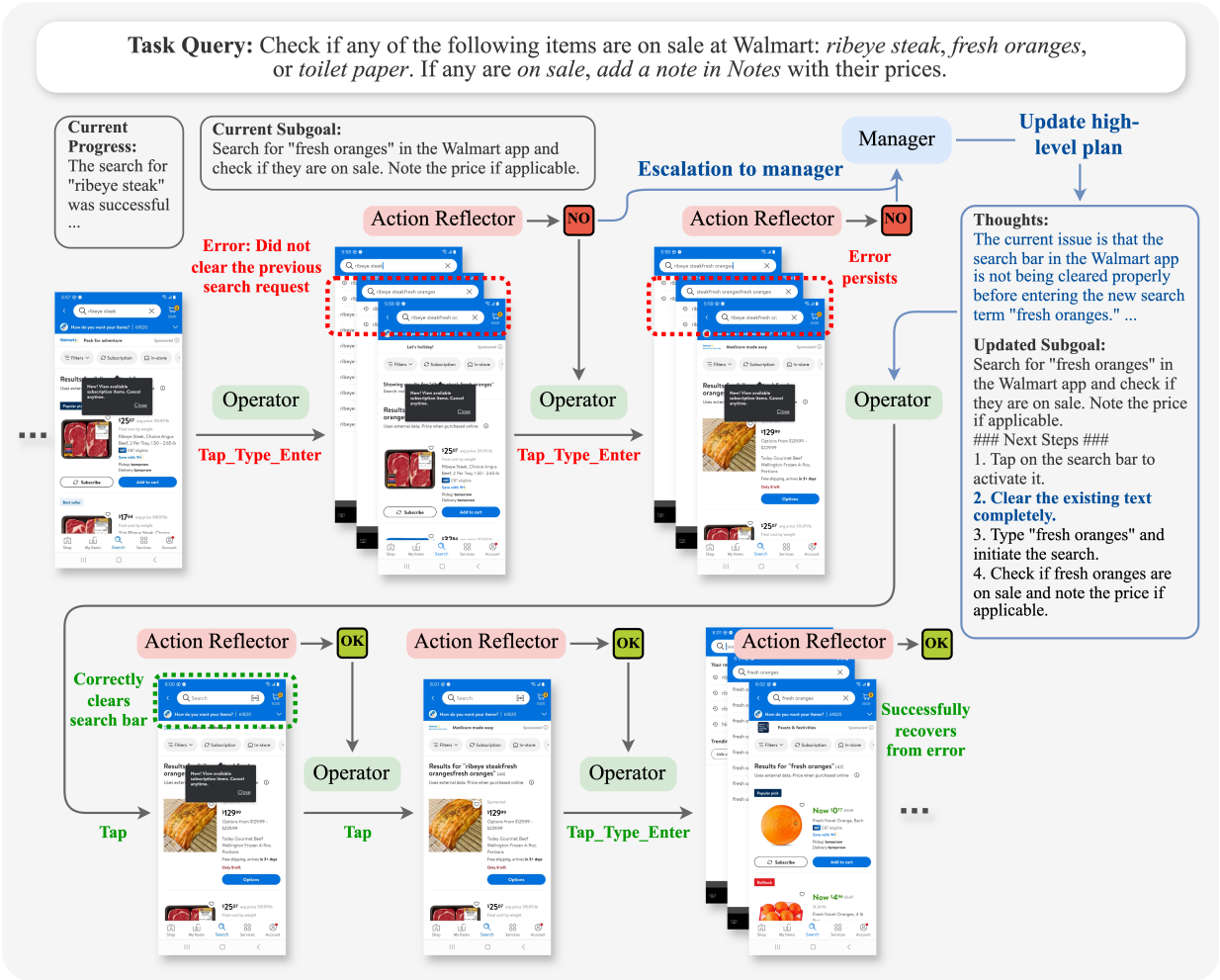


Figure 9. Error recovery with escalation. The task requires the agent to search for three different items on Walmart and note their sales information. At the step shown in the figure, the agent has already searched for ribeye steak and intends to search for fresh oranges next. However, the Operator erroneously calls the Shortcut that inputs text into the search bar and performs a search without clearing the previously entered text. Although the Action Reflector raises an error, the subgoal remains unchanged, and the Operator fails to rectify the error on the second attempt. After observing two consecutive errors, the error is escalated to the Manager, which correctly identifies the problem and revises the subgoal with detailed, decomposed steps to address the error. This helps the Operator correctly recover from the previous error by first tapping the "×" icon to clear the previous search query.

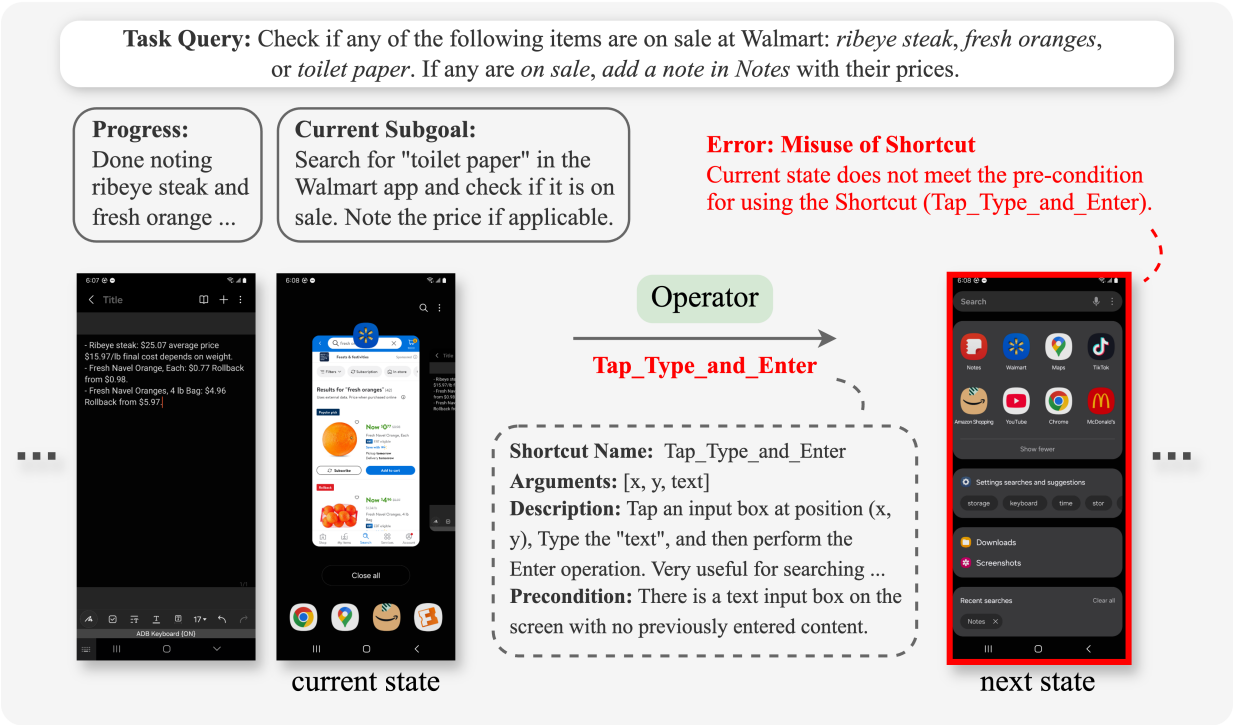


Figure 10. Example of misuse of Shortcuts in an invalid state. At the current step, as shown in the figure, the agent intended to switch back to Walmart to search for the final item requested by the user. While it correctly performs the "Switch_App" operation, it then calls a Shortcut for searching without realizing that it is not yet in the App where the search bar is available.

```
{
  "name": "Search_Location_in_Maps",
  "arguments": ["x","y","text"],
  "description": "Tap the search bar in Google Maps at position (x, y), type the location text, and select the first search result to display the route options.",
  "precondition": "The Google Maps app is open, and the search bar is visible on the screen.",
  "atomic_action_sequence": [
        {"name":"Tap","arguments_map":{"x":"x","y":"y"}},
        {"name":"Type","arguments_map":{"text":"text"}},
        {"name":"Enter","arguments_map":{}},
        {"name":"Tap","arguments_map":{"x":"x","y":"y"}}   <-- Redundant Tap action
  ]
}
```

```
{
  "name": "Switch_App_And_Search",
  "arguments": ["app_name","x","y","text"],
  "description": "Switch to a specified app, tap on a search bar at position (x, y), type the given text, and press Enter to perform a search.",
  "precondition": "The app to switch to is already open in the app switcher, and the search bar is visible on the screen after switching.",
  "atomic_action_sequence": [
        {"name":"Switch_App","arguments_map":{}},
        {"name":"Tap","arguments_map":{"x":"x","y":"y"}},   <-- Missing an additional Tap action to get into the App
        {"name":"Type","arguments_map":{"text":"text"}},
        {"name":"Enter","arguments_map":{}}
  ]
}
```

Figure 11. Example of imperfect (above) and erroneous (below) generated Shortcuts. The "Search_Location_in_Maps" Shortcut includes an unnecessary Tap action in the operation sequence, while the "Switch_App_And_Search" Shortcut omits a Tap action needed to first enter the desired App before performing the search.

Table 7. All task queries in Mobile-Eval-E.

Scenario	Task ID	APPs	Input Query
Restaurant Recommendation	1_late_night_korean_food	Maps	Find the best-rated late-night Korean restaurant in Champaign, IL that opens beyond 9pm on Google Maps.
	1_nearest_bakery	Maps	Get directions to the nearest Bakery that has a rating higher than 4.0 on Google Maps. Stop at the screen showing the route.
	1_thai_duck	Maps, Notes	Find the best-rated Thai restaurant in Urbana, IL that serves duck cuisine on Google Maps. Review customer comments and compile a summary of positive and negative feedback in Notes.
	1_bakery_birthday_cake	Maps, Notes	Find me a Bakery that is within 10min drive near me and does birthday cakes on Google Maps. Find the phone number and create a new note in Notes for that.
	1_chinese_ohare	Maps, X, Notes	Find me a popular Chinese restaurant near Chicago O'Hare airport on Google Maps. Check X for recent posts about their signature dishes and write a summary in Notes. Then get directions to that restaurant on Google Maps. Stop at the screen showing the route.
Information Researching	2_segment_anything_cited	Chrome	Find the most-cited paper that cites the paper 'Segment Anything' on Google Scholar. Stop at the screen showing the paper abstract.
	2_llm_agents_survey	Chrome, Notes	Find at least three representative survey papers on LLM agents on Google Scholar, and add their titles to the Notes.
	2_recipes_chinese	Chrome, YouTube	I have some onions, beef, and potatoes in my refrigerator. Can you find me a Chinese-style recipe that uses all three ingredients and can be prepared in under an hour? And find me a video tutorial on YouTube for that. Stop at the screen displaying the video.
	2_mcdonalds_deals	McDonald's, Maps	Can you check the McDonald's APP to see if there are any Rewards or Deals including Spicy McCrispy. If so, help me add that to Mobile Order (Do not pay yet, I will do it myself). And then check the pickup location and get directions on Google Maps. Stop at the screen showing the route.
	2_headphones_reviews	Amazon, Notes	Find three detailed user reviews of the Bose QC45 headphones from Amazon. Summarize the general sentiment in the Notes.
Online Shopping	3_oled_tv	Best Buy	Find the best deal on a 55-inch 4K OLED TV at Best Buy. Stop at the screen displaying the best deal you find.
	3_laptop_nvidia_gpu	Amazon Shopping	Find me a laptop on Amazon that is under $1000 with an Nvidia GPU and more than 8GB RAM.
	3_ninja_air_fryer	Amazon Shopping, Walmart	Compare the price of a Ninja air fryer 8 qt at Walmart and Amazon. Stop at the screen displaying the best deal you find.
	3_walmart_sale_items	Walmart, Notes	Check if any of the following items are on sale at Walmart: ribeye steak, fresh oranges, or toilet paper. If any are on sale, add a note in Notes with their prices.
	3_nintendo_switch_joy_con	Amazon Shopping, Best Buy, Walmart	I want to buy a brand-new Nintendo Switch Joy-Con. Any color is fine. Please compare the prices on Amazon, Walmart, and Best Buy. Find the cheapest option and stop at the screen where I can add it to the cart.
What's Trending	4_x_black_myth_wukong	X, Notes	Find the top posts about the game 'Black Myth Wukong' on X and summarize the key highlights in Notes.
	4_x_trending_news	X, Notes	Check the top 3 trending news on X. Read a few posts to figure out what's happening. And create a new Note to summarize your findings.
	4_watercolor_painting_tutorial	Lemon8, Notes	I want to learn how to paint watercolor. Find me some content creators to follow on Lemon8 that has highly liked posts about watercolor painting tutorials. List their account names in Notes.
	4_movie_trending	Fandango, Notes	Check the top 5 trending movies on Fandango that are currently in theaters. Compare their ratings and create a note in Notes for the highest-rated one, including its name and showtimes.
	4_horror_movie_reviews	Fandango, Lemon8, Notes	Find me the latest horror movie currently in theaters on Fandango. Check some reviews on Lemon8 about the movie and create a note in Notes with the general sentiment.
Travel Planning	5_cheap_flights_newyork	Booking	Find the cheapest round-trip flight from Chicago to New York City in the next month on Booking. Stop at the screen showing the best deal.
	5_things_to_do_la	Tripadvisor, Notes	Suggest some interesting things to do in LA. Find the top 3 attractions on Tripadvisor. Save the list in Notes.
	5_palo_alto_tour	Tripadvisor, Notes	Plan a one-day itinerary for Palo Alto, CA using Tripadvisor. Choose the attractions and dining recommendations, but keep in mind that I don't like seafood and I love museums. Write the plan in Notes.
	5_local_food_chicago	Tripadvisor, Notes	Find a highly recommended local restaurant in Chicago on Tripadvisor. Check the reviews about must-try dishes and summarize in Notes.
	5_hotel_champaign	Booking, Maps	Help me find a hotel in Champaign, IL on Booking that is under $200 for a queen bed. Make sure that the rating is higher than 7.0. Double-check on Google Maps to see if it is close to Green Street. Show me your final choice on Booking.

Table 8. Atomic operations space.

Operation	**Description**
$Open\_App(app\_name)$	If the current screen is Home or App screen, you can use this action to open the app named "app_name" on the visible on the current screen.
$Tap(x, y)$	Tap the position (x, y) in current screen.
$Swipe(x_1, y_1, x_2, y_2)$	Swipe from position (x_1, y_1) to position (x_2, y_2). To swipe up or down to review more content, you can adjust the y-coordinate offset based on the desired scroll distance. For example, setting $x_1 = x_2 = 0.5 * width$, $y_1 = 0.5 * height$, and $y_2 = 0.1 * height$ will swipe upwards to review additional content below. To swipe left or right in the App switcher screen to choose between open apps, set the x-coordinate offset to at least $0.5 * width$.
$Type(text)$	Type the "text" in an input box.
$Enter()$	Press the Enter key after typing (useful for searching).
$Switch\_App()$	Show the App switcher for switching between opened apps.
$Back()$	Return to the previous state.
$Home()$	Return to home page.
$Wait()$	Wait for 10 seconds to give more time for a page loading.

Inital Shortcuts (User Provided)


```
{
  "name": "Tap_Type_and_Enter",
  "arguments": ["x","y","text"],
  "description": "Tap an input box at position (x, y), Type the \"text\", and then perform the Enter operation. Very useful for searching and sending messages!",
  "precondition": "There is a text input box on the screen with no previously entered content.",
  "atomic_action_sequence": [{"name":"Tap","arguments_map":{"x":"x","y":"y"}},{"name":"Type","arguments_map":{"text":"text"}},{"name":"Enter","arguments_map":{}}]
}
```


Agent Generated Shortcuts


```
{
  "name": "Create_New_Note",
  "arguments": ["text"],
  "description": "Create a new note in the Notes app and type the provided text into it.",
  "precondition": "The Notes app is open, and the 'Add' button (orange icon with a pencil) is visible on the screen.",
  "atomic_action_sequence": [{"name":"Tap","arguments_map":{"x":"929","y":"2053"}},{"name":"Type","arguments_map":{"text":"text"}}]
}

{
  "name": "Search_Location_in_Maps",
  "arguments": ["x","y","text"],
  "description": "Tap the search bar in Google Maps at position (x, y), type the location text, and select the first search result to display the route options.",
  "precondition": "The Google Maps app is open, and the search bar is visible on the screen.",
  "atomic_action_sequence": [{"name":"Tap","arguments_map":{"x":"x","y":"y"}},{"name":"Type","arguments_map":{"text":"text"}},{"name":"Enter","arguments_map":{}},
{"name":"Tap","arguments_map":{"x":"x","y":"y"}}]
}

{
  "name": "Swipe_to_Reveal_Content",
  "arguments": ["x1","y1","x2","y2"],
  "description": "Swipe from position (x1, y1) to position (x2, y2) to reveal additional content below or above on the screen.",
  "precondition": "The screen contains content that can be revealed by swiping.",
  "atomic_action_sequence": [{"name":"Swipe","arguments_map":{"x1":"x1","y1":"y1","x2":"x2","y2":"y2"}}]
}

{
  "name": "Clear_Search_And_Type",
  "arguments": ["x_clear","y_clear","text"],
  "description": "Clear the current search term by tapping the 'X' icon and then type the new search term into the search bar.",
  "precondition": "The search bar is active, and the 'X' icon to clear the current search term is visible on the screen.",
  "atomic_action_sequence": [{"name":"Tap","arguments_map":{"x":"x_clear","y":"y_clear"}},{"name":"Type","arguments_map":{"text":"text"}}]
}

{
  "name": "Save_Note_As_File",
  "arguments": ["folder_x","folder_y","done_x","done_y","save_x","save_y"],
  "description": "Save a note as a file in a specified folder by selecting the folder, confirming the selection, and tapping the save button.",
  "precondition": "The 'Save note as' menu is open, and the desired folder, 'Done' button, and 'Save' button are visible on the screen.",
  "atomic_action_sequence": [{"name":"Tap","arguments_map":{"x":"folder_x","y":"folder_y"}},{"name":"Tap","arguments_map":{"x":"done_x","y":"done_y"}},
{"name":"Tap","arguments_map":{"x":"save_x","y":"save_y"}}]
}

{
  "name": "Switch_App_And_Search",
  "arguments": ["app_name","x","y","text"],
  "description": "Switch to a specified app, tap on a search bar at position (x, y), type the given text, and press Enter to perform a search.",
  "precondition": "The app to switch to is already open in the app switcher, and the search bar is visible on the screen after switching.",
  "atomic_action_sequence": [{"name":"Switch_App","arguments_map":{}},{"name":"Tap","arguments_map":{"x":"x","y":"y"}},{"name":"Type","arguments_map":
{"text":"text"}},{"name":"Enter","arguments_map":{}}]
}
```


Figure 12. Full list of Shortcuts generated by Mobile-Agent-E (with GPT-4o) after self-evolution.

**** Initial Tips (User Provided) ****

0. Do not add any payment information. If you are asked to sign in, ignore it or sign in as a guest if possible. Close any pop-up windows when opening an app. 1. By default, no apps are opened in the background. 2. Screenshots may show partial text in text boxes from your previous input; this does not count as an error. 3. When creating new Notes, you do not need to enter a title unless the user specifically requests it.

**** Agent Generated Tips (Scenario 1) ****

4. When searching for restaurants or businesses, ensure the query includes specific details like location, type of cuisine, and operational hours to narrow down results effectively. 5. Always verify the operational hours of businesses to ensure they meet the user's requirements, especially for late-night or time-sensitive searches. 6. When filtering search results (e.g., by rating or distance), ensure the filter criteria are applied correctly to avoid irrelevant results. 7. Double-check the selected location or business to ensure it matches the user's requirements (e.g., rating, proximity, or specific services offered) before proceeding to the route screen. 8. If the task involves creating a route, confirm that the route is displayed correctly and matches the intended destination before marking the subgoal as complete. 9. When navigating through menus or categories, use a systematic approach to ensure all relevant sections are explored thoroughly. 10. If an action does not return to the expected screen, use alternative navigation methods (e.g., tapping "X" or returning to the home screen) to correct the workflow. 11. When summarizing customer feedback, include both positive and negative aspects to provide a balanced overview. 12. When retrieving contact information, ensure the details (e.g., phone number or address) are accurate and match the selected business before saving them in Notes. 13. If a task involves multiple apps (e.g., Google Maps and Notes), ensure smooth transitions between apps and verify that the required information is correctly transferred. 14. If an app fails to open or respond in the app switcher, return to the home screen and reopen the app directly to avoid delays.

**** Agent Generated Tips (Scenario 2) ****

4. When identifying the most-cited paper or similar tasks, ensure to sort the results by citation count if the option is available. This minimizes manual scanning and reduces errors. 5. If a search action fails, verify the input text and ensure the correct search bar is targeted before retrying. Adjust the tap location if necessary. 6. When recording information from search results, ensure the details are accurate and clearly formatted to avoid confusion. 7. If a task involves multiple steps across different apps, always confirm the completion of one step before proceeding to the next to avoid errors or omissions. 8. If a search query fails to execute, double-check the tap location and ensure the search bar is properly activated before typing. Retry the action with slight adjustments if necessary. 9. When selecting a video or item from a list, ensure the title matches the intended choice to avoid selecting the wrong option. 10. If a button or option does not respond to a tap, ensure it is fully visible on the screen. Use a swipe or scroll action to adjust the view if necessary before retrying. 11. When switching between apps, ensure the correct app is selected from the app switcher to avoid unnecessary navigation errors. 12. Always stop at the final screen requested by the user, ensuring the task is fully completed before ending the interaction.

**** Agent Generated Tips (Scenario 3) ****

4. When identifying the best deal, prioritize both price and features, and ensure any discounts or promotions are clearly noted. 5. Always confirm that the displayed product matches the search criteria (e.g., size, specifications) to avoid selecting an incorrect item. 6. If the task requires stopping at a specific screen, ensure the screen is fully loaded and all relevant details are visible before stopping. 7. If a filter does not apply correctly, try adjusting it again by swiping or tapping alternative areas of the screen to reveal hidden options. 8. When using sliders for filters (e.g., price range), swiping is often more effective than tapping to adjust the values. 9. If a filter unexpectedly resets or removes itself, reapply it and verify the results before proceeding. 10. Always double-check the final results to ensure all filters (e.g., price, specifications) have been applied correctly. 11. When comparing prices across platforms, ensure that the product model and specifications (e.g., size, features) are identical to avoid inaccurate comparisons. 12. If swiping to reveal content, ensure the swipe is smooth and covers enough distance to load all relevant details on the screen. 13. If an app fails to open or navigate correctly, return to the home screen and retry the action. This often resolves navigation issues. 14. If a tap action does not work as expected, consider tapping alternative areas of the screen, such as associated buttons or options, to achieve the desired outcome. 15. When switching between apps, ensure the correct app is reopened and verify the screen before proceeding to avoid unnecessary repetition.

**** Agent Generated Tips (Scenario 4) ****

4. When navigating apps, ensure that the correct icon is tapped by carefully identifying its position and function to avoid misalignment or unintended actions. 5. If a search filter is applied unintentionally, clear it by tapping the "X" icon in the search bar before proceeding with a new search. 6. Always verify the context of the search results to ensure they align with the intended query before summarizing or proceeding to the next step. 7. When recording information in Notes, ensure the formatting is clear and consistent for easy readability. 8. Double-check the accuracy of the recorded information (e.g., account names, titles) before saving the note to avoid errors. 9. If redirected to an unintended page (e.g., "My Orders"), navigate back to the main interface or intended section before proceeding. 10. When comparing multiple items (e.g., movie ratings), keep track of all relevant data to ensure accurate comparisons and avoid revisiting the same pages unnecessarily. 11. If an app opens an unintended interface (e.g., camera instead of Notes), return to the home screen and retry opening the correct app to avoid confusion. 12. When entering search terms, ensure the previous query is cleared completely to prevent appending incorrect text to the new query. 13. If a misaligned tap opens an unintended menu (e.g., Filters), close it immediately and retry the intended action. 14. Use broader search terms if specific queries fail to yield results, and refine the search gradually based on the context. 15. If an app fails to execute a search or action, consider switching to a browser or alternative app to complete the task.

**** Agent Generated Tips (Scenario 5) ****

4. Always confirm that the displayed results match the search criteria (e.g., correct cities, dates, and round-trip selection) before proceeding to the next step. 5. If multiple options are displayed, ensure the cheapest or most relevant option is clearly identified and selected as per the task requirements. 6. If a "Back" button fails to function as expected, consider alternative methods to save or exit, such as using a menu or additional options (e.g., "Save as file"). 7. When saving a note as a file, ensure the correct folder and file format are selected before confirming the save. 8. Double-check that the task is fully completed (e.g., the note is saved in the correct location) before marking it as done. 9. If scrolling through content does not reveal new information, consider alternative methods to locate the required details, such as using a search or filter function within the app. 10. If the end of a section is reached and the required information is not found, reassess the search criteria or explore other sections of the app for relevant details. 11. When searching for specific items (e.g., dishes, amenities), use keywords or filters to narrow down results and save time. 12. If repetitive actions (e.g., swiping) fail to yield results, pause and evaluate whether the task can be completed using a different approach or if the information is unavailable. 13. When switching between apps, ensure that the context of the task is maintained, and verify that the information gathered in one app aligns with the requirements in the other app. 14. Always confirm the proximity or location details (e.g., using Google Maps) before finalizing a selection, especially when location is a key criterion.

Figure 13. Full list of Tips generated by Mobile-Agent-E (with GPT-4o) after self-evolution.