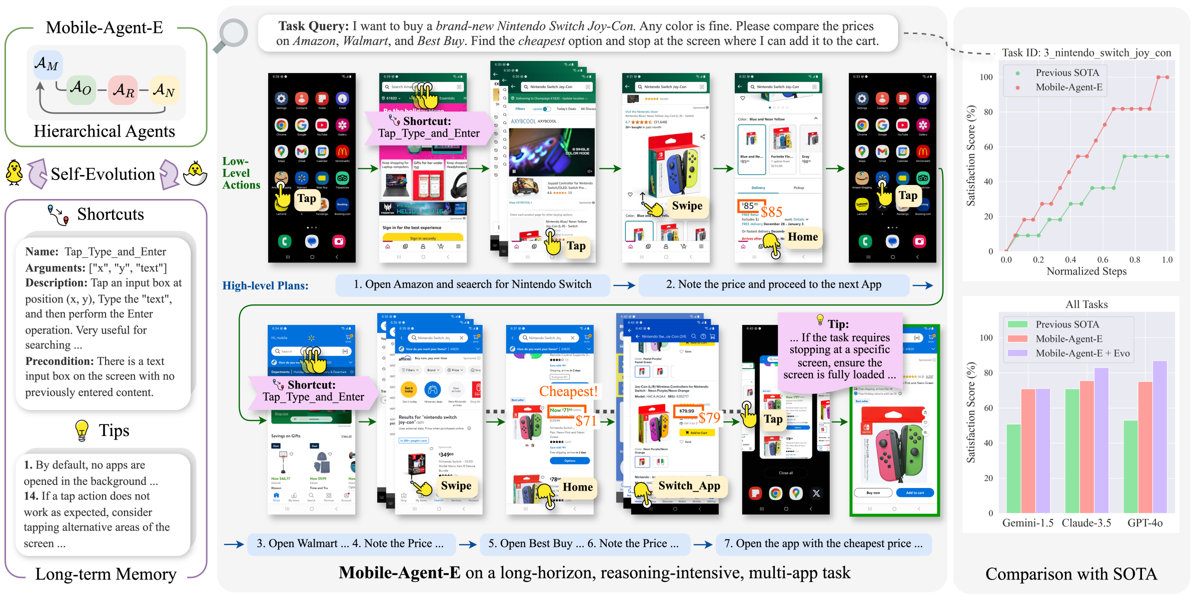
Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks

Anonymous ACL submission



Task Query: 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.

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 ...
Precondition: There is a text input box on the screen with no previously entered content.

1. By default, no apps are opened in the background ...
14. If a tap action does not work as expected, consider tapping alternative areas of the screen ...

Tip: ... If the task requires stopping at a specific screen, ensure the screen is fully loaded ...

1. Open Amazon and seaerch for Nintendo Switch → 2. Note the price and proceed to the next App → 3. Open Walmart ... 4. Note the Price ... → 5. Open Best Buy ... 6. Note the Price ... → 7. Open the app with the cheapest price ...

Figure 1: Introducing Mobile-Agent-E, a novel hierarchical agentic framework designed for complex real-world mobile tasks. Mobile-Agent-E disentangles high-level planning from low-level actions, significantly outperforming previous state-of-the-art approaches (Zhang et al., 2023; Wang et al., 2024b,a). Equipped with a novel self-evolution module that learns general *Tips* and reusable *Shortcuts* from past experiences, Mobile-Agent-E further enhances both performance and efficiency.

Abstract

Recent advancements in large multimodal model (LMM)-based mobile agents have demonstrated promising capabilities in acting within mobile environments. However, current approaches face significant limitations: (1) they fall short in addressing real-world human needs, which involve complex, open-ended, and reasoning-intensive tasks; and (2) they lack mechanisms to learn and improve from prior experiences. To address these challenges, we introduce **Mobile-Agent-E**, a hierarchical agentic framework capable of self-evolution through past experience. Mobile-Agent-E adopts a multi-level communication protocol for reasoning, perception, and error recovery, explicitly separating high-level planning from low-level action decisions. It also introduces a novel self-evolution module that maintains a persistent long-term memory comprising *Tips* and *Shortcuts*, enabling continual refinement of task performance and efficiency. To bridge the gap in existing benchmarks for complex, open-ended tasks, we further present a new benchmark—**Mobile-Eval-E**—alongside a new evaluation metric, the Satisfaction Score. Empirical results show that Mobile-Agent-E achieves a 22% absolute improvement over previous state-of-the-art approaches across three LMM backbones. We also provide a comprehensive analysis of the impact of the self-evolution mechanism and outline promising directions for future work.

1 Introduction

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 acting in the Web, PC, and mobile environments. 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, existing mobile agents and benchmarks focus primarily on *goal-oriented* tasks, such as "Create a new contact for {name}. Their number is {number}" (Rawles et al., 2024). These tasks typically follow a linear ground-truth trajectory and have a single success state. However, we argue that tasks more representative of real human needs are significantly more complex. They often require: **(1) intensive reasoning** to satisfy multiple constraints; **(2) long-horizon planning** that spans across multiple apps; and **(3) open-ended exploration**, where vague instructions demand active information gathering. For instance, as illustrated in Figure1, online shopping often involves navigating across different apps to compare prices and find the best deal.

Second, unlike humans who quickly adapt to recurring tasks on new devices, current mobile agents lack the ability to learn from prior experiences. They treat every task as if it were their first attempt, allocating the same computational resources at each step and repeating the same mistakes. This inability to accumulate knowledge from past experience significantly limits both their effectiveness and efficiency on complex, long-horizon tasks, where subroutines such as searching or creating notes are frequently reused across different objectives.

To address these limitations, we propose **Mobile-Agent-E**, a **hierarchical agentic framework** capable of **self-evolution** through past experiences. Mobile-Agent-E explicitly separates high-level planning—such as decomposing tasks into subgoals—from low-level action decisions like determining specific actions and their parameters (e.g., `tap(x,y)`), enabling multi-level reasoning, perception, and error recovery. Figure 1 shows a demo of Mobile-Agent-E on a challenging online shopping task. Mobile-Agent-E also features a novel self-evolution module, which includes a persistent long-term memory containing two types of critical knowledge: *Tips* and *Shortcuts*. This design draws inspiration from human cognitive science, where Tips are akin to the lessons encoded in episodic memory (Tulving, 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 and Zola, 1996; Anderson, 1982).

To address the limitation of existing mobile benchmarks, which mainly include goal-oriented tasks, we introduce **Mobile-Eval-E**, a new challenging benchmark focusing on complex, open-ended, real-world tasks. Mobile-Eval-E features more than twice the number of expected operations

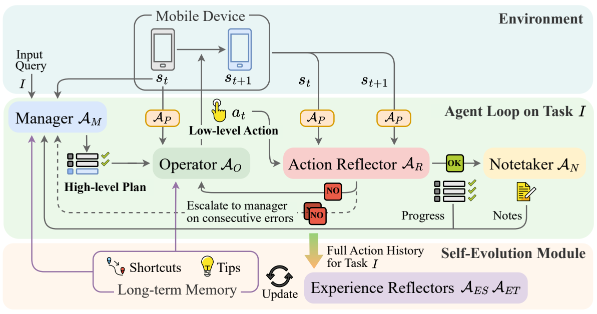


Figure 2: An overview of Mobile-Agent-E.

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 multi-app tasks. Accompanying the benchmark, we introduce Satisfaction Score, a new metric to address the challenge posed by real-world tasks that often lack a binary success flag or a ground truth trajectory. This evaluation offers a reliable measure of agent performance aligned with real-world human needs.

To conclude, our contributions are threefold:

- **Mobile-Eval-E Benchmark**: A shift in focus from goal-oriented tasks to complex, open-ended, real-world mobile tasks.
- **Mobile-Agent-E Framework**: A hierarchical agent architecture featuring multi-level reasoning and error recovery, achieving a 22.1% absolute improvement over prior state-of-the-art approaches.
- **Self-Evolution Module**: The first work to explore self-evolution in mobile agents, yielding a 6.5% improvement in performance and a 12.9% gain in efficiency. Comprehensive analysis provides further insights for future research.

2 Mobile-Agent-E

Figure 2 provides an overview of Mobile-Agent-E. We detail the hierarchical agentic framework (§2.1) and the self-evolution module (§2.2) below.

2.1 Hierachical Agentic Framework

Multi-Level Reasoning. The key idea behind improving a model's performance on complex tasks is to disentangle high-level planning from low-level actions. Specifically, we divide all reasoning agents into two levels: the **planning level**, which contains a `Manager`, and the **action level**, which contains an `Operator`, an `Action Reflector`, and a `Notetaker`. **Figure 3** provides a detailed breakdown of the inputs and outputs for each agent. All reasoning agents are instantiated from a frozen large multimodal model (LMM), such as GPT-

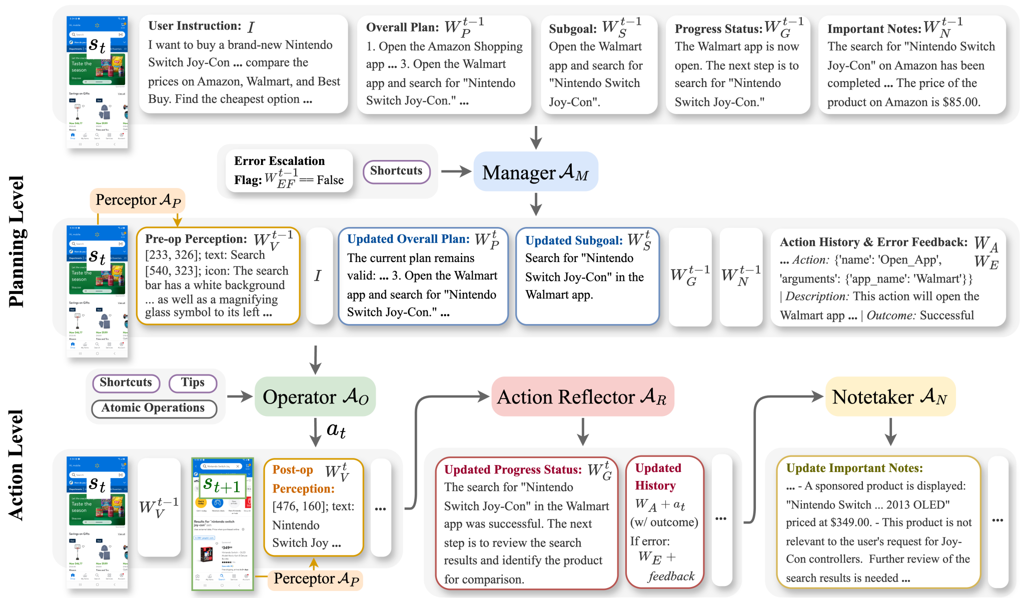

Planning Level
User Instruction: I
I want to buy a brand-new Nintendo Switch Joy-Con ... compare the prices on Amazon, Walmart, and Best Buy. Find the cheapest option ...
Overall Plan: W_P^{t-1}
1. Open the Amazon Shopping app ... 3. Open the Walmart app and search for "Nintendo Switch Joy-Con." ...
Subgoal: W_S^{t-1}
Open the Walmart app and search for "Nintendo Switch Joy-Con".
Progress Status: W_G^{t-1}
The Walmart app is now open. The next step is to search for "Nintendo Switch Joy-Con."
Important Notes: W_N^{t-1}
The search for "Nintendo Switch Joy-Con" on Amazon has been completed ... The price of the product on Amazon is $85.00.
Error Escalation Flag: W_{EF}^{t-1} == False
Shortcuts
Manager $\mathcal{A}_M$
Perceptor $\mathcal{A}_P$
s_t
Pre-op Perception: W_V^{t-1}
[233, 326]; text: Search [540, 323]; icon: The search bar has a white background ... as well as a magnifying glass symbol to its left ...
I
Updated Overall Plan: W_P^t
The current plan remains valid: ... 3. Open the Walmart app and search for "Nintendo Switch Joy-Con." ...
Updated Subgoal: W_S^t
Search for "Nintendo Switch Joy-Con" in the Walmart app.
W_G^{t-1}
W_N^{t-1}
Action History & Error Feedback: W_A W_E
... Action: {'name': 'Open_App', 'arguments': {'app_name': 'Walmart'}} | Description: This action will open the Walmart app ... | Outcome: Successful
Action Level
Shortcuts
Tips
Atomic Operations
Operator $\mathcal{A}_O$
a_t
Action Reflector $\mathcal{A}_R$
Notetaker $\mathcal{A}_N$
s_t
W_V^{t-1}
s_{t+1}
Post-op Perception: W_V^t
[476, 160]; text: Nintendo Switch Joy ...
Perceptor $\mathcal{A}_P$
...
Updated Progress Status: W_G^t
The search for "Nintendo Switch Joy-Con" in the Walmart app was successful. The next step is to review the search results and identify the product for comparison.
Updated History
$W_A + a_t$ (w/ outcome)
If error: $W_E +$ feedback
...
Update Important Notes:
... - A sponsored product is displayed: "Nintendo Switch ... 2013 OLED" priced at $349.00. - This product is not relevant to the user's request for Joy-Con controllers. Further review of the search results is needed ...
...

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.

Notation	**Description**
Environment	
I	Input task query
a^t	Action at step t
s^t	Phone state (screenshot)
Working Memory	
W_V^t	Fine-grained visual perception
W_P^t	Overall plan (subgoals)
W_S^t	Current subgoal
W_G^t	Progress status
W_N^t	Important notes
W_{EF}^t	Error Escalation Flag
$\mathbf{W_A}$	Action history with outcome status
$\mathbf{W_E}$	Error history with feedback
Long-term Memory	
L_S	Shortcuts
L_T	Tips

Table 1: Summary of notations for intermediate inputs and outputs. Notations of agents are defined in §2.1.

4o (OpenAI, 2024). We formally define each agent below, with **notations given in Table 1**.

The **Manager** ($\mathcal{A}_M$) focuses on devising high-level plans, i.e., identifying overall strategies and the next immediate subgoals, to fulfill the user's request. Note that the Shortcuts L_S (detailed in §2.2) are also provided to the Manager to guide efficient high-level planning.

$$W_P^t, W_S^t = \mathcal{A}_M(I, s_t, W_P^{t-1}, W_S^{t-1}, W_G^{t-1}, W_N^{t-1}, L_S)$$

The **Operator** ($\mathcal{A}_O$) decides which concrete action to perform based on the high-level plans from the Manager and the latest m-step history.[*] The Operator also considers the *Tips* as guidance from the long-term memory, which can be self-evolved from past experiences. Unlike the Manager, the action level agents take the fine-grained perception results W_V^t—in addition to the screenshot s_t—as input. We detail the perception module later in this section.

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

The action space of a_t 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 9. We detail the definitions and examples of *Shortcuts* and *Tips* in §2.2.

The **Action Reflector** ($\mathcal{A}_R$) 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

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

[†] 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.

$\mathbf{W_A}[t]$ 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 $\mathbf{W_E}[t]$.

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

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`** ($\mathcal{A}_N$) is dedicated to extracting and aggregating task-relevant information W_N^t after each step.

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

Multi-Level Perception. The reasoning agents at different levels require different granularities of perception of the current phone state. At the *planning level*, the Manager only needs a holistic visual context of the screen to decide on high-level plans; therefore, we provide only the screenshot s_t to the Manager. At the *action level*, however, the agents need precise coordinates of interactive elements to predict and verify actions. To address this, we introduce the **`Perceptor`** **($\mathcal{A}_P$)**, 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 components: an OCR model, an icon-grounding model, and an icon-captioning model. Given a screenshot s_t, the Perceptor produces a fine-grained list of texts and icons along with their corresponding coordinates W_V^t.

Multi-Level Error Recovery. The ability to recover from errors is particularly important for executing complex, long-horizon tasks. In addition to the two-level reasoning agents, we introduce a two-level error-recovery mechanism that operates at both the action and planning levels. When an error first occurs (as reported by the Action Reflector), the Operator attempts to address it at the action level, for example, by tapping a different location. If the model becomes stuck in an error loop, i.e., it observes k consecutive failed actions (e.g., $k = 2$), a special Error Escalation Flag, W_{EF}^{t-1}, is raised to the Manager. In this case, the Manager receives additional information about the recent errors, $\mathbf{W_E}[-k:]$, and is asked to determine how to address the issue from a higher-level perspective, such as refining the overall plan or adjusting the current subgoal. A concrete example of how error escalation aids recovery is shown in Figure 9.

2.2 Self-Evolution Module

Inspired by how humans quickly adapt to new tasks, 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 model.

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.

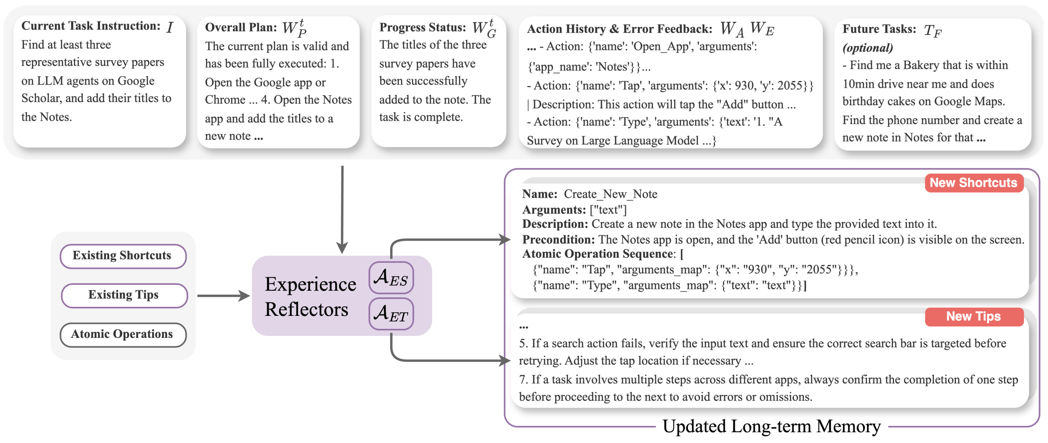
Shortcuts (L_S) are defined as reusable, executable functions composed of sequences of atomic operations tailored for recurring subroutines. Shortcuts are akin to procedural knowledge, which allows humans to perform well-practiced tasks efficiently and often subconsciously (Squire and Zola, 1996; Anderson, 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 (Figure 1) 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 based on the interaction history and optionally a list of future tasks T_F. The Experience Reflectors are also instantiated from frozen LMMs/LLMs. **Figure 4** provides a detailed breakdown of one self-evolution step. Figures 12 and 13 shows a full list of generated Shortcuts and Tips by Mobile-Agent-E.

3 Mobile-Eval-E Benchmark

3.1 Towards Complex, Open-Ended, Real-World Tasks

Existing mobile benchmarks—based on either simulated environments (Rawles et al., 2024; Chen et al., 2024) or dynamic actual devices (Wang et al., 2024b; Zhang et al., 2023; Wang et al., 2024a)—primarily focus on goal-oriented tasks. These tasks


Current Task Instruction: I
Find at least three representative survey papers on LLM agents on Google Scholar, and add their titles to the Notes.

Overall Plan: W_P^t
The current plan is valid and has been fully executed: 1. Open the Google app or Chrome ... 4. Open the Notes app and add the titles to a new note ...

Progress Status: W_G^t
The titles of the three survey papers have been successfully added to the note. The task is complete.

Action History & Error Feedback: W_A W_E
... - Action: {'name': 'Open_App', 'arguments': {'app_name': 'Notes'}}...
- Action: {'name': 'Tap', 'arguments': {'x': 930, 'y': 2055}} | Description: This action will tap the "Add" button ...
- Action: {'name': 'Type', 'arguments': {'text': '1. "A Survey on Large Language Model ...}

Future Tasks: T_F
(optional)
- 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 **...**

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 (red pencil icon) is visible on the screen.
Atomic Operation Sequence: **[**
```
  {"name": "Tap", "arguments_map": {"x": "930", "y": "2055"}}},
  {"name": "Type", "arguments_map": {"text": "text"}}]
```

...
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 ...
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.

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.

Benchmark	#Tasks	#Multi-App Tasks	#Apps	Avg # Steps	Total # Steps
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

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

often have a ground-truth trajectory and a unique end-state. In real-world scenarios, however, such tasks are unrealistic and leave a gap between benchmarking performance and practical applications.

To address this limitation, we propose **Mobile-Eval-E** benchmark, which emphasizes complex, open-ended, real-world tasks. Mobile-Eval-E is designed for dynamic evaluation on actual devices and comprises 25 manually crafted tasks spanning five realistic scenarios: "Restaurant Recommendation," "Information Searching," "Online Shopping," "What's Trending," and "Travel Planning." Mobile-Eval-E tasks are long-horizon and reasoning-intensive, often admitting multiple satisfactory trajectories and success states. As shown in Table 2, Mobile-Eval-E significantly surpasses previous dynamic benchmarks in complexity, featuring more than **2**× the number of expected steps per task. Additionally, Mobile-Eval-E encompasses a broader range of apps, with 76% of tasks requiring interactions with multiple apps. In §4, we demonstrate that this benchmark poses a substantial challenge for existing state-of-the-art models. The full set of task queries can be found in Appendix Table 8.

3.2 Fine-Grained Evaluation Metrics

Previous benchmarks typically employ a *Binary Success Rate (BS)* or a completion rate against a "ground-truth" trajectory to evaluate task completeness. However, the complexity and open-endedness of Mobile-Eval-E tasks pose unique challenges in faithfully assessing model performance. For example, many 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 to ground-truth states.

For each task, we manually write a list of rubrics (an example is shown in Figure 5(a)), containing both milestone steps (e.g., "Opened Tripadvisor") and exploration criteria (e.g., "Viewed multiple attractions"). We then introduce 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)* to evaluate action-level performance, and a *Termination Error (TE)* rate to reflect the agent's robustness and error-recovery capability. Details about the termination modes can be found in Appendix B. To keep the human evaluation workload reasonable for this fine-grained analysis, we maintain a relatively small number of tasks in Mobile-Eval-E.

4 Experiments

We consider two evaluation settings—*without* and *with* evolution—to comprehensively assess both the hierarchical agentic framework and the self-evolution module. Our primary focus is on com-

Model	Type	Binary Success Rate (%) ↑	Satisfaction Score (%) ↑	Action Acc (%) ↑	Reflection Acc (%) ↑	Termination Error (%) ↓
Traditional Evaluation Without Evolution						
AppAgent (Zhang et al., 2023)	Single-Agent	0.0	25.2	60.7	-	96.0
Mobile-Agent-v1 (Wang et al., 2024b)	Single-Agent	4.0	45.5	69.8	-	68.0
Mobile-Agent-v2 (Wang et al., 2024a)	Multi-Agent	8.0	53.0	73.2	96.7	52.0
Mobile-Agent-E (ours)	Multi-Agent	44.0	75.1	85.9	97.4	32.0
Evaluation With Cross-Task Evolution						
Mobile-Agent-E + Evo (ours)	Multi-Agent	40.0	86.9	90.4	97.8	12.0

Table 3: Comparison with state-of-the-art models on *complex open-ended tasks* in Mobile-Eval-E. GPT-4o is used as the LMM backbone for all methods.

Model	Gemini-1.5-pro					Claude-3.5-Sonnet					GPT-4o				
	BS↑	SS↑	AA↑	RA↑	TE↓	BS↑	SS↑	AA↑	RA↑	TE↓	BS↑	SS↑	AA↑	RA↑	TE↓
Traditional Evaluation Without Evolution															
Mobile-Agent-v2 (Wang et al., 2024a)	4.0	50.8	63.4	83.9	64.0	12.0	70.9	76.4	96.9	32.0	8.0	53.0	73.2	96.7	52.0
Mobile-Agent-E (ours)	20.0	70.9	74.3	91.3	48.0	32.0	75.5	91.1	99.1	12.0	44.0	75.1	85.9	97.4	32.0
Evaluation With Cross-Task Evolution															
Mobile-Agent-E + Evo (ours)	20.0	71.2	77.4	89.6	48.0	40.0	83.0	91.4	99.7	12.0	40.0	86.9	90.4	97.8	12.0

Table 4: Results on different LMM backbones, including GPT-4o, Gemini, and Claude. Metrics are defined in §3.2.

Model	Success Rate (Pass@1 %)
Traditional Evaluation Without Evolution	
T3A + GPT-4-turbo (Rawles et al., 2024)	30.6
M3A + GPT-4-turbo (Rawles et al., 2024)	25.4
Ponder & Press + GPT-4o (Wang et al., 2024d)	34.5
Aria-UI + GPT-4o (Yang et al., 2024)	44.8
UGround + GPT-4o (Gou et al., 2025)	44.0
Mobile-Agent-E + GPT-4o (ours)	45.0

Table 5: Comparison with state-of-the-art models on traditional *goal-oriented tasks* in Android World. We compare with methods released before Feb 2025.

plex, reasoning-intensive tasks from Mobile-Eval-E. We also evaluate on Android World (Rawles et al., 2024), which comprises 116 goal-oriented tasks across diverse apps. More details on baselines and model implementation can be found in Appendix B.

Traditional evaluation without evolution. In this setting, we evaluate each task individually without any cross-task information sharing. For Mobile-Eval-E, we perform dynamic, real-time evaluation (Wang et al., 2024b; Zhang et al., 2023; Wang et al., 2024a) on a physical device. Specifically, we use the Android Debug Bridge (ADB) to control an Android phone[‡] and conduct human evaluation on the recorded screenshots and action histories. For Android World, we follow the official setup in an Android emulator[§], where evaluation is automated by verifying the final state. We adopt the same screen representation as M3A, including screenshots and the accessibility (A11y) tree.

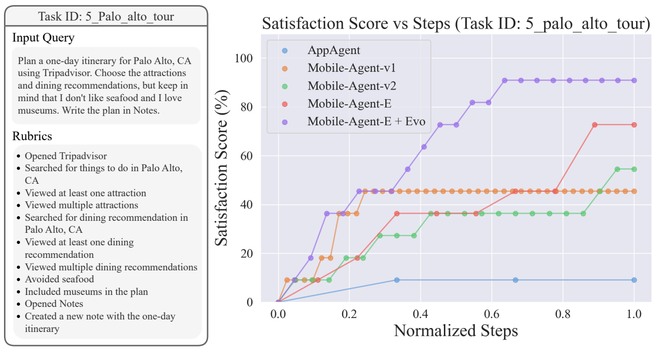
Evaluation with cross-task evolution. To our knowledge, this is the first work to explore a cross-task evolution evaluation. In this setting, an agent is given a group of tasks and executes them sequentially, while maintaining a *persistent long-term memory* across tasks. Specifically, for Mobile-Eval-E, we form five groups corresponding to the five scenarios, each containing five tasks. We evaluate Mobile-Agent-E with the self-evolution module enabled (referred to as **Mobile-Agent-E + Evo**). 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 controlled setting allows us to investigate what is important for the agent to evolve from past experience (i.e., Tips and Shortcuts), and how this accumulated knowledge will impact subsequent tasks.

5 Results

Comparison with state-of-the-art. Tables 3 and 5 show that Mobile-Agent-E significantly outperforms prior SOTA (22.1%) on complex open-ended tasks, while also setting a new SOTA on traditional goal-oriented tasks in Android World. This comparison particularly highlights the effectiveness of the hierarchical agentic framework. Our approach also demonstrates superior robustness and

[‡]We use Samsung Galaxy A15.

[§]`https://github.com/google-research/android_world`


Task ID: 5_Palo_alto_tour

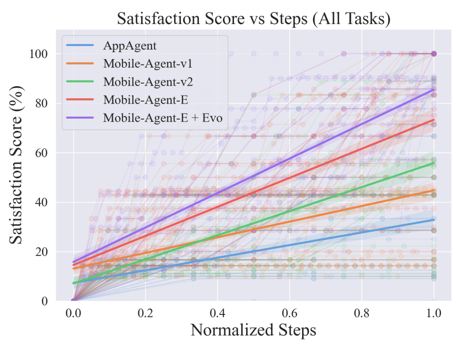
Input Query

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.

Rubrics

- Opened Tripadvisor
- Searched for things to do in Palo Alto, CA
- Viewed at least one attraction
- Viewed multiple attractions
- Searched for dining recommendation in Palo Alto, CA
- Viewed at least one dining recommendation
- Viewed multiple dining recommendations
- Avoided seafood
- Included museums in the plan
- Opened Notes
- Created a new note with the one-day itinerary

(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 present the human-written rubrics for the task. In (b), we additionally include a linear regression line for each model. To enable visualization of trajectories with different 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. A steeper and higher line indicates better efficiency and effectiveness.

error recovery capabilities, as indicated by a significantly lower Termination Error rate. Moreover, enabling self-evolution further enhances performance, leading to an improvement of 6.5% against no evolution. In §5.1, we provide further analysis of the evolution module.

Varying reasoning backbones. Table 4 demonstrates that Mobile-Agent-E can bring consistent improvements on all recent LMMs, including GPT-4o, Claude-3.5-Sonnet, and Gemini-1.5-pro. Moreover, we observe that self-evolution yields greater benefits when paired with stronger backbones.

Satisfaction Score v.s. Binary Success Rate. As shown in Tables 3 and 4, the Binary Success Rates (BS) are sparse and exhibit large jumps between different models. The Satisfaction Score (SS), on the other hand, provides a smoother, more faithful measure of the models' performance.

Task completion efficiency. Evaluating the efficiency of mobile agents on complex, 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 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. As shown in Figure 5, we observe that Mobile-Agent-E not only achieves better final performance but also fulfills rubrics faster.

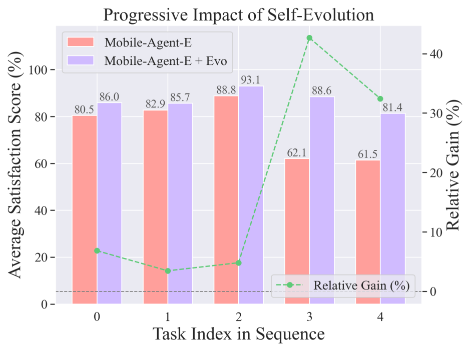


Figure 6: Later tasks in self-evolution show greater improvements, reflecting the growing impact of iterative evolution. The scores are averaged over five scenarios.

5.1 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 evolution. In Figure 6, the x-axis reflects the task index in the sequence it is performed. We observe a generally increasing trend indicating that the gain tends to be more significant in later tasks.

Shortcuts reduce 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. Note that when the self-evolution module is disabled, Mobile-Agent-E is provided with fixed initial Tips and a single example Shortcut. In Table 6, we observe a positive correlation between using more Shortcuts and

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

Table 6: 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 effectively accelerates inference, achieving comparable times to previous, simpler frameworks.

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

Table 7: Unique impact from the evolved Tips.

faster inference speed. This is because a Shortcut enables the execution of multiple operations within a single decision-making iteration.

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 ablate on 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 evolved Tips. Table 7 shows that Tips alone serve as an important aspect of self-evolution.

Managing self-evolution in the long run. To explore how to efficiently manage a large number of Tips and Shortcuts accumulated over the long term, we further include a case study in Appendix A. This case study introduces mechanisms to retrieve only the most relevant information for a new task, ensuring the agent remains effective even as its experience base grows significantly.

6 Related Work

6.1 GUI Agents

The advancement of large multimodal models (LMM) has driven research on LMM-based GUI agents (Wang et al., 2024c), focusing on AI assistants for GUI environments like 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). For mobile, research has enhanced single-agent perception and reasoning via tool usage (Wang et al., 2024b) and exploration (Zhang et al., 2023; Li et al., 2024). Recent multi-agent systems (Rawles et al., 2024; Wang et al., 2024a) show promise but still face challenges like short-sighted planning and poor error recovery.

It is worth noting that the "planning" module in Mobile-Agent-v2 (Wang et al., 2024a) merely serves as a progress tracker and is *fundamentally different* from the proposed Manager in Mobile-Agent-E. In Mobile-Agent-v2, the "decision-making" module remains responsible for both high-level planning (e.g., "what to do next") and low-level action execution (e.g., "where to tap"), resulting in a flat and overloaded design. In contrast, Mobile-Agent-E introduces a hierarchical agentic structure that explicitly separates high-level planning from low-level action decisions.

6.2 Self-Evolution in Foundation Models

Self-improvement in large language and multimodal models has been widely explored (Tao et al., 2024), through techniques like iterative refinement (Madaan et al., 2024), self-reflection (Shinn et al., 2024), self-training (Huang et al., 2022), and multi-persona collaboration (Wang et al., 2023). Recent work also emphasizes tool learning and creation (Cai et al., 2023; Qian et al., 2023; Yuan et al., 2023). In GUI agents, self-evolution is less explored. While Cradle (Tan et al., 2024) shows potential in skill curation for PC environments, how to do evolution in mobile settings remains unaddressed. In this work, we identify two important types of knowledge for evolution, namely Tips and Shortcuts.

7 Conclusion and Future Work

In this work, we make the first attempt to build mobile agents for complex, real-world tasks, demonstrating the effectiveness of hierarachical agentic framework as well as self-evolution. Future work will focus on developing improved strategies for generating, invoking, and revising long-term memory, as well as automating the evaluation of complex mobile tasks. Detailed discussions are included in the Limitations section.

Limitations

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.

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.

Human evaluation is required for Mobile-Eval-E. Due to the complexity and open-ended nature of these real-world tasks, no current multimodal model can replace human evaluation for providing a reliable assessment. In the future, as multimodal reasoning models advance, we believe it is possible to develop an automatic rubric verifier that determines whether each criterion is met at a given step based on the global action history. We leave this promising direction to future work.

Broader Impacts

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

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

Anthropic. 2024. Claude 3.5 Sonnet.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A Frontier Large Vision-Language Model with Versatile Abilities. *arXiv preprint arXiv:2308.12966*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, and 1 others. 2024. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *NeurIPS 2024 Workshop on Open-World Agents*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2023. Cogagent: A visual language model for gui agents. *Preprint*, arXiv:2312.08914.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*.

Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. 2020. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11474–11481.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. 2023. Grounding DINO: marrying DINO with grounded pre-training for open-set object detection. *CoRR*, abs/2303.05499.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, and 1 others. 2024a. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, and 1 others. 2024b. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, and 1 others. 2024. Gui agents: A survey. *arXiv preprint arXiv:2412.13501*.

OpenAI. 2024. GPT-4o System Card.

Cheng Qian, Chi Han, Yi R Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. *arXiv preprint arXiv:2305.14318*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

Revanth Gangi Reddy, Sagnik Mukherjee, Jeonghwan Kim, Zhenhailong Wang, Dilek Hakkani-Tur, and Heng Ji. 2024. Infogent: An agent-based framework for web information aggregation. *arXiv preprint arXiv:2410.19054*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

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

Weihao Tan, Ziluo Ding, Wentao Zhang, Boyu Li, Bohan Zhou, Junpeng Yue, Haochong Xia, Jiechuan Jiang, Longtao Zheng, Xinrun Xu, and 1 others. 2024. 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*.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

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

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv preprint arXiv:2406.01014*.

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

Shuai Wang, Weiwen Liu, Jingxuan Chen, Weinan Gan, Xingshan Zeng, Shuai Yu, Xinlong Hao, Kun Shao, Yasheng Wang, and Ruiming Tang. 2024c. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*.

Yiqin Wang, Haoji Zhang, Jingqi Tian, and Yansong Tang. 2024d. Ponder & press: Advancing visual gui agent towards general computer control. *arXiv preprint arXiv:2412.01268*.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2024. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. Assistantbench: Can web agents solve realistic and time-consuming tasks? *Preprint*, arXiv:2407.15711.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R Fung, Hao Peng, and Heng Ji. 2023. Craft: Customizing llms by creating and retrieving from specialized toolsets. *arXiv preprint arXiv:2309.17428*.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. UFO: A UI-Focused Agent for Windows OS Interaction. *arXiv preprint arXiv:2402.07939*.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *Preprint*, arXiv:2312.13771.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*.

A Case Study: Towards 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 59 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 I and H.

B Experimental Details

Baselines. For Mobile-Eval-E, we compare against diverse open-sourced mobile agent frameworks compatible with actual devices, 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. AppAgent-v2 (Li et al., 2024) is not included since it is not open-sourced at the time of writing.

We also compare with a wider range of models with reported scores on Android World, such as M3A (Rawles et al., 2024), Aria-UI (Yang et al., 2024) and UGround (Gou et al., 2025). We further explore using different large multimodal models (LMM) as backbones for the reasoning agents, including GPT-4o (OpenAI, 2024), Claude-3.5-Sonnet (Anthropic, 2024), and Gemini-1.5-pro (Team et al., 2024). Unless otherwise specified, the default backbone for all models is GPT-4o.

Backbone versions. The detailed versions of the large multimodal models are listed as follows: (1) GPT-4o version: gpt-4o-2024-11-20; (2) Claude-3.5 version: claude-3-5-sonnet-20241022; (3) Gemini-1.5 version: gemini-1.5-pro-latest (Dec 2024)

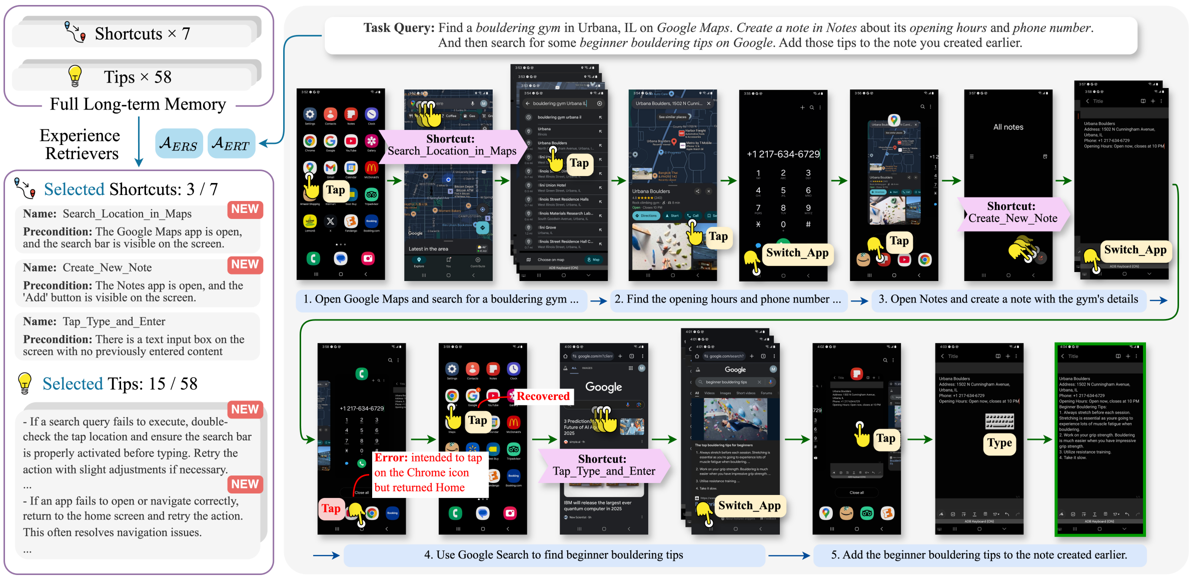
Perceptor implementation details. We follow Mobile-Agent-v2 (Wang et al., 2024a) to implement the Perceptor in Mobile-Agent-E with slight modifications. We use DBNet[¶](Liao et al., 2020) and ConvNextViT-document[‖] from ModelScope for OCR detection and recognition respectively. We use GroundingDINO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

Agent termination modes. 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 (TE). The TE rate is computed as the ratio of tasks with termination errors to all tasks.

Note that we recognize self-reported success may be inaccurate; however, in the TE metric we treat it as a normal termination. Early termination is already penalized via a low satisfaction score, while the TE metric is designed to capture the model's robustness to truly abnormal terminations—such as infinite loops, formatting errors, and so on.

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

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



Name: Search_Location_in_Maps
Precondition: The Google Maps app is open, and the search bar is visible on the screen.

Name: Create_New_Note
Precondition: The Notes app is open, and the 'Add' button is visible on the screen.

Name: Tap_Type_and_Enter
Precondition: There is a text input box on the screen with no previously entered content

- 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.

...

- If an app fails to open or navigate correctly, return to the home screen and retry the action. This often resolves navigation issues.

...

Task Query: Find a *bouldering gym* in Urbana, IL on *Google Maps*. *Create a note in Notes* about its *opening hours* and *phone number*. And then search for some *beginner bouldering tips on Google*. Add those tips to the note you created earlier.

1. Open Google Maps and search for a bouldering gym ... → 2. Find the opening hours and phone number ... → 3. Open Notes and create a note with the gym's details →

→ 4. Use Google Search to find beginner bouldering tips → 5. Add the beginner bouldering tips to the note created earlier.

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.

C 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.

D Impact of Multi-level Error Recovery

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.

E Illustration of Remaining Limitations

Figure 10 illustrates an example of a misuse of Shortcuts in an invalid state. Figure 11 illustrates an example of erroneous and imperfect Shortcuts.

F All Tasks in Mobile-Eval-E Benchmark

Table 8 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.

G Atomic Operation Space

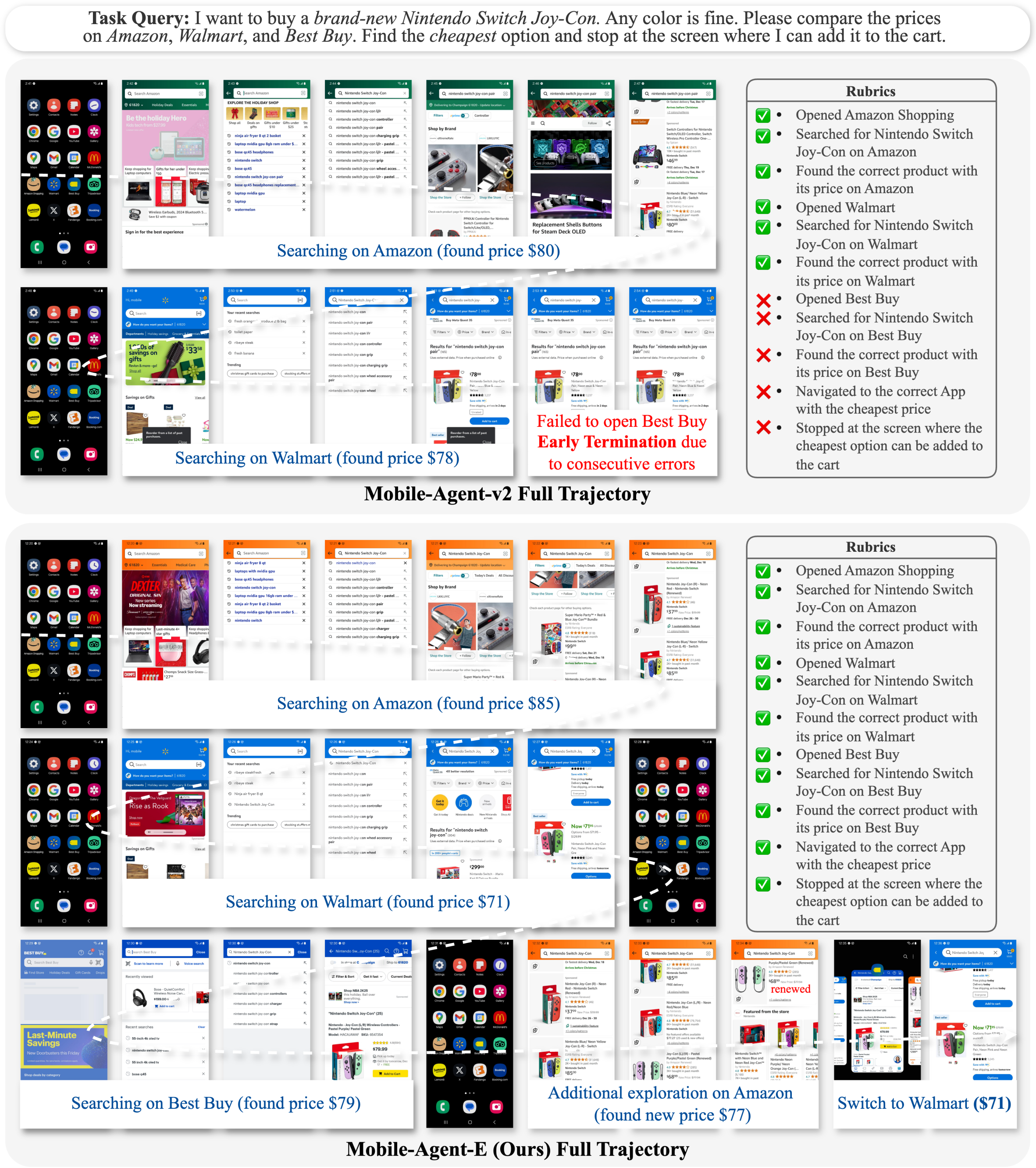
Table 9 presents all atomic operations considered in Mobile-Agent-E.

H 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.

I 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.


Task Query: 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.

Mobile-Agent-v2 Full Trajectory

Rubrics

- ✓ Opened Amazon Shopping
- ✓ Searched for Nintendo Switch Joy-Con on Amazon
- ✓ Found the correct product with its price on Amazon
- ✓ Opened Walmart
- ✓ Searched for Nintendo Switch Joy-Con on Walmart
- ✓ Found the correct product with its price on Walmart
- ✗ Opened Best Buy
- ✗ Searched for Nintendo Switch Joy-Con on Best Buy
- ✗ Found the correct product with its price on Best Buy
- ✗ Navigated to the correct App with the cheapest price
- ✗ Stopped at the screen where the cheapest option can be added to the cart

Mobile-Agent-E (Ours) Full Trajectory

Rubrics

- ✓ Opened Amazon Shopping
- ✓ Searched for Nintendo Switch Joy-Con on Amazon
- ✓ Found the correct product with its price on Amazon
- ✓ Opened Walmart
- ✓ Searched for Nintendo Switch Joy-Con on Walmart
- ✓ Found the correct product with its price on Walmart
- ✓ Opened Best Buy
- ✓ Searched for Nintendo Switch Joy-Con on Best Buy
- ✓ Found the correct product with its price on Best Buy
- ✓ Navigated to the correct App with the cheapest price
- ✓ Stopped at the screen where the cheapest option can be added to the cart

Searching on Amazon (found price $80)
Searching on Walmart (found price $78)
Failed to open Best Buy
Early Termination due to consecutive errors
Searching on Amazon (found price $85)
Searching on Walmart (found price $71)
Searching on Best Buy (found price $79)
Additional exploration on Amazon (found new price $77)
Switch to Walmart ($71)
renewed

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

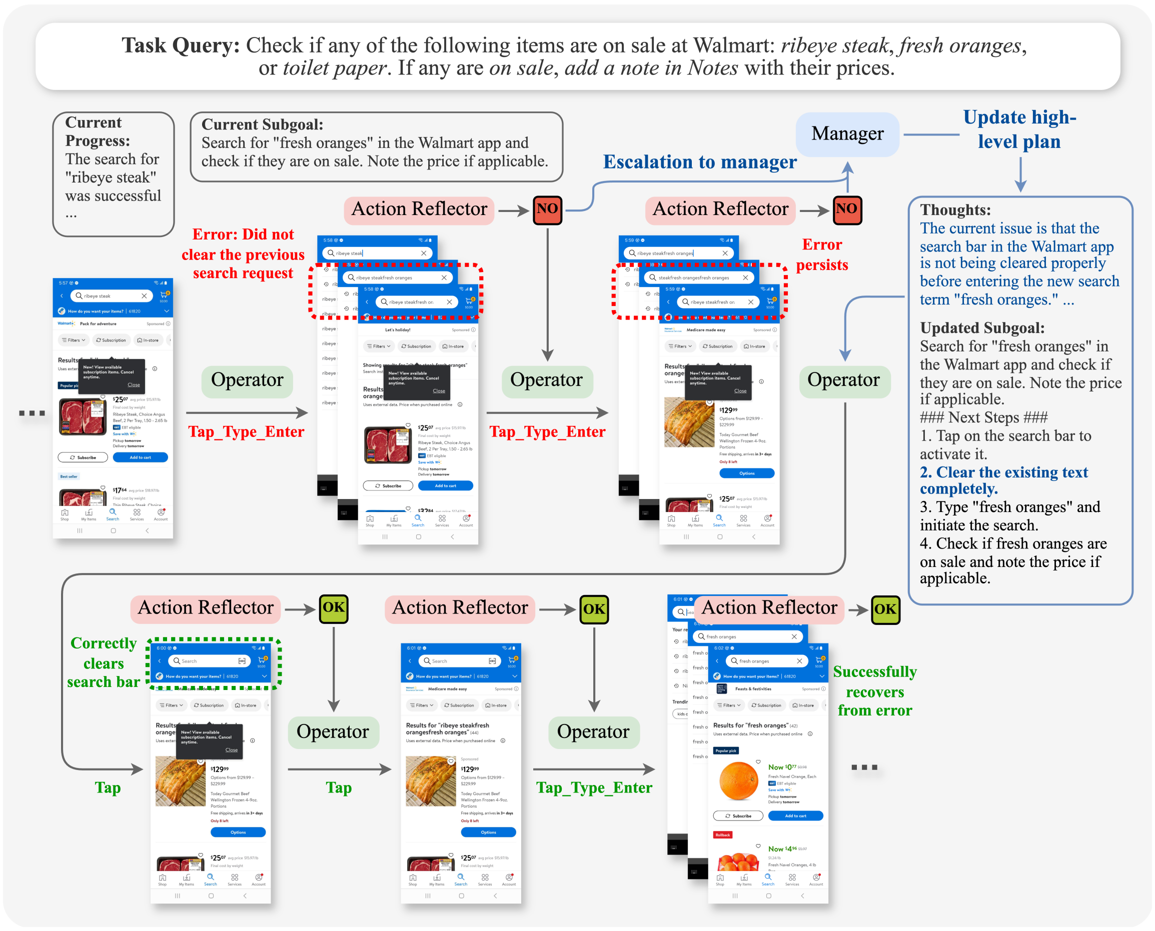
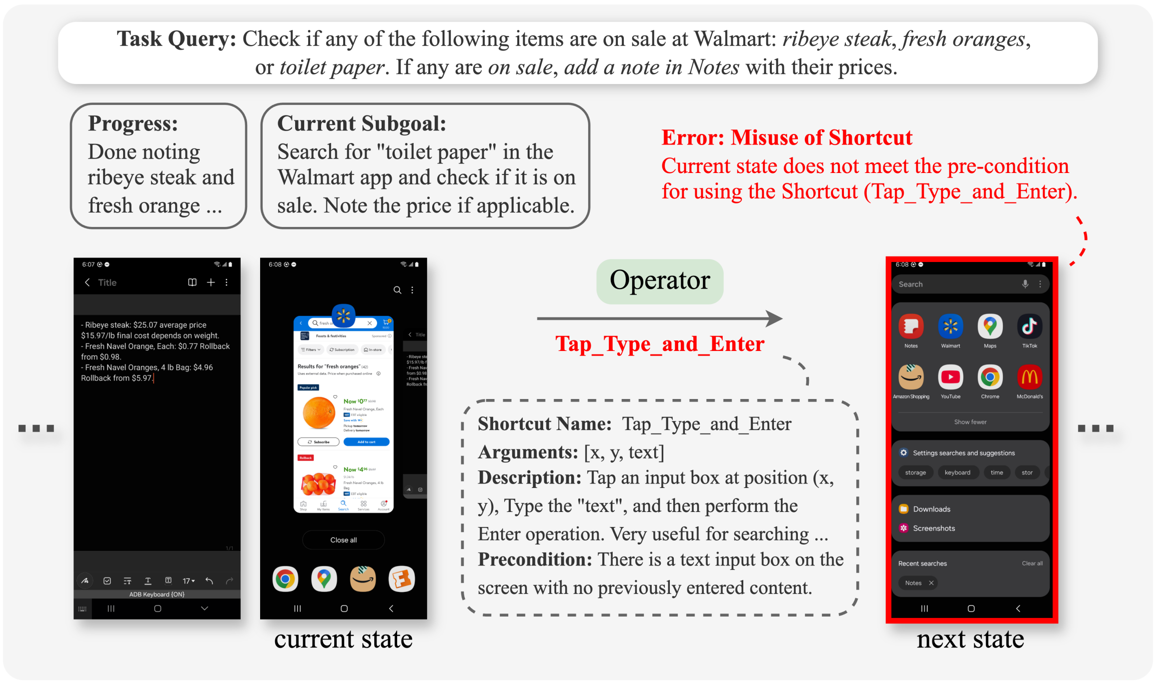

Task Query: Check if any of the following items are on sale at Walmart: ribeye steak, fresh oranges, or toilet paper. If any are on sale, add a note in Notes with their prices.
Current Progress: The search for "ribeye steak" was successful ...
Current Subgoal: Search for "fresh oranges" in the Walmart app and check if they are on sale. Note the price if applicable.
Manager
Update high-level plan
Escalation to manager
Action Reflector
NO
Error: Did not clear the previous search request
Error persists
Operator
Tap_Type_Enter
Thoughts: The current issue is that the search bar in the Walmart app is not being cleared properly before entering the new search term "fresh oranges." ...
Updated Subgoal: Search for "fresh oranges" in the Walmart app and check if they are on sale. Note the price if applicable.
### Next Steps ###
1. Tap on the search bar to activate it.
2. Clear the existing text completely.
3. Type "fresh oranges" and initiate the search.
4. Check if fresh oranges are on sale and note the price if applicable.
OK
Correctly clears search bar
Tap
Successfully recovers from error
...

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.



Task Query: 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.

Progress: Done noting ribeye steak and fresh orange ...

Current Subgoal: Search for "toilet paper" in the Walmart app and check if it is on sale. Note the price if applicable.

Error: Misuse of Shortcut
Current state does not meet the pre-condition for using the Shortcut (Tap_Type_and_Enter).

Shortcut 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 ...
Precondition: There is a text input box on the screen with no previously entered content.

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":{}},          Missing an additional Tap
        {"name":"Tap","arguments_map":{"x":"x","y":"y"}},   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.

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: All task queries in Mobile-Eval-E.

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.

Table 9: Atomic operations space.

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.