WebPilot: A Versatile and Autonomous Multi-Agent System for Web Task Execution with Strategic Exploration

Yao Zhang[1,3*†], **Zijian Ma**[2*], **Yunpu Ma**[1,3†], **Zhen Han**[1], **Yu Wu**[2], **Volker Tresp**[1,3†]
[1]LMU Munich, Munich, Germany
[2]Technical University of Munich, Munich, Germany
[3]Munich Center for Machine Learning (MCML), Munich, Germany
yzhang@dbs.ifi.lmu.de, tresp@dbs.ifi.lmu.de

Abstract

LLM-based autonomous agents often fail to execute complex web tasks that require dynamic interaction, largely due to the inherent uncertainty and complexity of these environments. Existing LLM-based web agents typically rely on rigid, expert-designed policies specific to certain states and actions, lacking the flexibility and generalizability needed to adapt to unseen tasks. In contrast, humans excel by exploring unknowns, continuously adapting strategies based on new observations, and resolving ambiguities through exploration. To emulate human-like adaptability, web agents need strategic exploration and complex decision-making. Monte Carlo Tree Search (MCTS) is well-suited for this, but classical MCTS struggles with vast action spaces, unpredictable state transitions, and incomplete information in web tasks. In light of this, we develop WebPilot, a multi-agent system with a dual optimization strategy that improves MCTS to better handle complex web environments. Specifically, the Global Optimization phase involves generating a high-level plan by breaking down tasks into manageable subtasks, continuously refining this plan through reflective analysis of new observations and previous subtask attempts, thereby focusing the search process and mitigating challenges posed by vast action spaces in classical MCTS. Subsequently, the Local Optimization phase executes each subtask using a tailored MCTS designed for complex environments, effectively addressing uncertainties and managing incomplete information by iteratively refining decisions based on new observations. Experimental results on WebArena and MiniWoB++ demonstrate the effectiveness of WebPilot. Notably, on WebArena, WebPilot achieves SOTA performance with GPT-4, achieving a **93% relative increase** in success rate over the concurrent tree search-based method. WebPilot advances autonomous agents, enabling more reliable decision-making in practical environments.

Introduction

The advanced reasoning capabilities of LLMs (Yang et al. 2023; Achiam et al. 2023; Team et al. 2023; Anthropic 2024) have expanded the potential for autonomous web agents capable of navigating and interacting in complex, dynamic environments (Lai et al. 2024; Deng et al. 2024). To succeed, these agents must handle tasks like complex information retrieval, long-horizon task execution, and integrating diverse information sources (Wang et al. 2024; Zhou et al. 2023b).

However, despite the advanced reasoning capabilities of LLMs, current LLM-based web agents (Sodhi et al. 2024; Ma et al. 2023) often fall short in executing complex web tasks that require dynamic interaction. This limitation arises primarily from their heavy reliance on rigid, expert-designed policies tailored to specific states and actions. While these policies are meticulously crafted to address well-defined scenarios, they inherently lack the flexibility and generalizability needed to adapt to the uncertain and variable nature of real-world web environments, as well as to unseen tasks.

In contrast, humans excel at handling complex web tasks due to their cognitive flexibility (Daw, Niv, and Dayan 2005), enabling them to explore unknowns, adjust plans dynamically based on new observations, and resolve ambiguities through trial and error. Monte Carlo Tree Search (MCTS) mirrors this process, facilitating exploration and refining actions iteratively based on feedback, much like humans. It adjusts tactical-level strategies during the search process, refining action generation at each node based on the feedback from the expansion, much like how humans iteratively adjust their actions in response to new observations. When encountering dead ends or unclear paths—nodes with low potential or uncertain outcomes—MCTS reassesses and explores alternative branches, effectively addressing the limitations of LLMs in handling unfamiliar web environments.

Despite its potential, classical MCTS (Browne et al. 2012) struggles in complex web environments due to vast action spaces, unpredictable state transitions, and incomplete information. While recent methods like LLM-MCTS (Zhao, Lee, and Hsu 2024) and LATS (Zhou et al. 2023a) integrate LLMs for heuristic guidance, they are limited to tasks with lower complexity, reducing their effectiveness in real-world scenarios. Similarly, RAP (Hao et al. 2023a) optimizes inference paths but lacks the flexibility for dynamic interaction in complex environments. Reward mechanism within MCTS also remains challenging for complex environments; current approaches either rely on direct environment reward (Zhou et al. 2023a), which is impractical for real-world tasks, or use overly simplistic scoring systems (e.g., binary or low-resolution scales (Koh et al. 2024)), failing to accurately capture the nuanced and evolving nature of web environments.

*These authors contributed equally.
†Corresponding authors

Benchmark	Dynamic Interaction	Partial Obs. Env.	Non-Fixed Policy	Scalable	Realistic Web Env.	Comp. Self-Reward
RAP (Hao et al. 2023b)	✗	✗	✗	✗	✗	✗
LATS (Zhou et al. 2023a)	✓	✗	✗	✗	✗	✗
LLM-MCTS (Zhao, Lee, and Hsu 2024)	✗	✓	✗	✓	✗	✗
SteP (Sodhi et al. 2024)	✓	✓	✗	✗	✓	✗
LM-Tree Search (Koh et al. 2024)	✓	✓	✓	✗	✓	✗
WebPilot	✓	✓	✓	✓	✓	✓

Table 1: Comparison of different agent types, including web agents and MCTS-based agents. Partial Obs. Env. - Partially Observable Environment. Comp. Self-Reward - Comprehensive self-reward mechanism.

In response, we introduce WebPilot, a versatile multi-agent system designed with a dual optimization strategy grounded in the principles of MCTS, specifically tailored for enhanced adaptability in complex environments. WebPilot first applies Global Optimization, decomposing tasks and refining high-level plans through reflective analysis. This enables the system to dynamically adapt to evolving objectives while effectively managing the complexities of vast action spaces. Following this, WebPilot employs Local Optimization to execute each subtask using a customized MCTS approach, addressing uncertainties and incomplete information by iteratively refining decisions based on new observations.

Specifically, the Global Optimization phase is driven by *Planner*, *Controller*, and *Extractor*. It begins with Hierarchical Task Decomposition (HTD), where *Planner* breaks down complex tasks into manageable subtasks, narrowing the focus and effectively addressing the vast action spaces that challenge classical MCTS. Reflective Task Adjustment (RTA) then refines plans based on new observations, enabling dynamic adaptation. *Controller* monitors subtask progression, assessing subtask completeness and generating reflections for re-execution when needed. Meanwhile, *Extractor* gathers essential information to support task execution. This coordinated approach ensures WebPilot remains adaptable and efficient in dynamic environments.

For each subtask, WebPilot employs Local Optimization strategies involving *Explorer*, *Verifier*, *Appraiser*, and *Controller* to enhance execution in dynamic environments. Goal-Oriented Selection (GOS) leverages LLM intuitions to steer toward promising states, while Reflection-Enhanced Node Expansion (RENE) refines tactical-level strategies using real-time feedback. Dynamic Evaluation and Simulation (DES) assesses actions and simulates outcomes, and Maximal Value Backpropagation (MVB) prioritizes paths with the highest potential. Together with Global Optimization, these strategies ensure adaptable and efficient task execution, enabling WebPilot to outperform existing web agents.

Experiments on MiniWoB++ (Liu et al. 2018) and WebArena (Zhou et al. 2023b) are chosen to assess the performance of WebPilot in environments with varying complexity. The results highlight the superiority of WebPilot, particularly in the complex, realistic web environment. In WebArena, WebPilot achieves an impressive 37.2% success rate, surpassing the current SOTA method, SteP (Sodhi et al. 2024), which relies on rigid, expert-designed policies tailored to specific states and actions. Notably, WebPilot demonstrates a remarkable 93% relative increase in success rate over concurrent tree-based methods (Koh et al. 2024). Even when using GPT-3.5, WebPilot remains highly competitive with GPT-4-based SOTA methods, achieving a 29.1% success rate. These results underscore the exceptional ability of WebPilot to handle the uncertainty and complexity inherent in real-world web environments.

The primary contributions of this work are as follows:

1. We introduce WebPilot, an autonomous multi-agent system designed for complex web environments, combining global and local MCTS-inspired optimization strategies to enable human-like flexibility in exploration, adaptation, and decision-making at both the subtask and action levels.

2. We develop a Hierarchical Reflection Mechanism, incorporating Strategic Reflection in Global Optimization and Tactical Reflection in Local Optimization, which significantly enhances decision-making in evolving environments.

3. We introduce a novel Granular Bifaceted Self-Reward Mechanism that guides MCTS by integrating action effectiveness with goal-oriented potential, allowing for more precise assessments in dynamic and ambiguous environments.

4. WebPilot achieves SOTA performance, particularly in challenging benchmarks like WebArena, demonstrating substantial advancements in general autonomous agent capabilities for complex real-world tasks.

Related Work

We provide a comparative analysis of LLM-based web agents and MCTS-based agents, with details in Tab. 1.

LLM-Based Autonomous Web Agents

Recent advancements in LLMs have enabled the development of web agents that leverage LLM reasoning to interact with web environments. One approach (Kim, Baldi, and McAleer 2024; Sun et al. 2024; Prasad et al. 2023; Fu et al. 2024; Zheng et al. 2023) relies on environment-specific state-action pairs embedded in demonstrations to respond to specific observations, as seen in SteP (Sodhi et al. 2024), which utilizes rigid, expert-designed policies tailored to specific states and actions. However, these agents struggle to adapt to unseen tasks. Another approach (Li et al. 2023; Zhou et al. 2023a; Pan et al. 2024; Koh et al. 2024) adopts a strategy of freely exploring and discovering unknown environments. Despite efforts like Auto Eval & Refine (Pan et al. 2024), which integrates an evaluator into Reflexion (Shinn

et al. 2024), and LM-Tree Search (Koh et al. 2024), which employs a search-based method in realistic environments, these agents still face challenges with complex tasks, leaving room for significant advancements. In contrast, WebPilot employs a dual optimization strategy, excelling at task exploration and dynamic strategy adjustments, enabling superior adaptability in complex environments.

LLM-MCTS Applications

MCTS, originally developed for the game of Go (Coulom 2006; Browne et al. 2012), is renowned for its effectiveness in handling exploration problems. Enhanced by the Upper Confidence bounds applied to Trees (UCT) method (Kocsis and Szepesvári 2006), MCTS has found extensive use in fields such as robotics(Zhao, Lee, and Hsu 2024), strategy games (Jang et al. 2021), and autonomous vehicles (Lenz, Kessler, and Knoll 2016). Recently, researchers have integrated LLMs with MCTS to tackle various NLP tasks, including QA (Hong et al. 2023; Xie et al. 2024; Chi, Yang, and Klein 2024), prompt refinement techniques (Wang et al. 2023), and complex mathematical reasoning problems (Tian et al. 2024; Zhang et al. 2024). Building on this integration, LLM-based agents have also incorporated MCTS to enhance their exploratory and decision-making capabilities. For instance, LATS (Zhou et al. 2023a) applies MCTS to simple web tasks. However, traditional MCTS-based methods often encounter difficulties in scenarios with vast action spaces, unpredictable state transitions, and incomplete information in web tasks. WebPilot addresses these challenges by utilizing a tailored MCTS designed specifically for complex environments, effectively navigating and optimizing decision-making processes even in highly uncertain situations.

Methodology

Promblem Formulation

Our objective is to enable the LLM-based web agent to effectively solve a task $\mathcal{T}$ in a web environment $\mathcal{E}$ by emulating human web navigation strategies. Web environments are partially observable, limiting the information available to agents and complicating problem-solving. This occurs because web content changes dynamically, preventing the agent from fully anticipating or knowing the state of elements—such as updated content or availability—until interaction. Following WebArena (Zhou et al. 2023b), we use an accessibility tree (actree) to represent observations, capturing the structure and interactive elements of the web page. However, due to the lack of specific web domain knowledge, LLMs often struggle to recognize or utilize the functionalities of various web elements. Thus, the agent must actively explore the environment to gather critical information about the task and the web elements, making informed decisions despite these uncertainties and incomplete information.

This process is modeled as a Partially Observable Markov Decision Process (POMDP), where the environment $\mathcal{E}$ is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, and an observation space $\mathcal{O}$. The transition function $\mathcal{F} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ governs state changes based on actions, typically in a deterministic manner. Task execution requires the agent to make decisions based on partial observations o_t at each time step t, where action a_t leads to a new state s_{t+1} and observation o_{t+1}. The evaluation function $eval(\boldsymbol{a}, \boldsymbol{s})$, defined by the environment, determines task success by assessing whether the sequence of executed actions $\boldsymbol{a} = \{a_1, \ldots, a_n\}$ and states $\boldsymbol{s} = \{s_1, \ldots, s_n\}$ satisfy the task $\mathcal{T}$.

Global Optimization: Adaptive Strategy Refinement through Reflective Adjustment

The Global Optimization phase emulates human cognition by leveraging prior knowledge to generate an initial plan for unfamiliar tasks. However, due to the lack of specific web domain knowledge in LLMs and the dynamic, uncertain nature of web environments, this initial plan misses critical details and struggles to remain effective as the environment evolves. To address this, WebPilot continuously refines the initial plan through reflective analysis of new observations and previous subtask outcomes. Global Optimization involves two key components: Hierarchical Task Decomposition (HTD) and Reflective Task Adjustment (RTA), which are facilitated by the *Planner*, *Controller*, and *Extractor*.

Hierarchical Task Decomposition (HTD) begins with *Planner* breaking down complex tasks into manageable subtasks $\mathcal{T}_i$, thereby creating a flexible high-level plan that can adapt to the uncertain and ever-changing conditions of web environments. In generating this plan, *Planner* utilizes only a few high-level demonstrations to ensure robust and adaptive task decomposition. This approach allows WebPilot to dynamically adjust its strategies for specific aspects of each task, making it more responsive to environmental changes. Unlike the concurrent search-based web agent (Koh et al. 2024), which struggles with complex tasks due to its expanding search space, HTD ensures that each subtask is more targeted and efficient. This decomposition enables WebPilot to refine each subtask in real time, adjusting dynamically to evolving conditions without requiring a complete overhaul of the entire task execution. By concentrating on these manageable subtasks, WebPilot employs MCTS-enhanced decision strategies, specifically through the Local Optimization phase, to minimize unnecessary search paths and optimize decision-making within a focused scope, effectively mitigating the challenges posed by vast action spaces that often hinder classical MCTS. The effectiveness of *Planner* is demonstrated through ablation studies.

Reflective Task Adjustment (RTA) Upon completing each subtask in the Local Optimization phase, WebPilot reassesses and refines its high-level plan to ensure alignment with the overall task $\mathcal{T}$. Guided by *Controller* and *Planner*, this process critically evaluates the execution of each subtask against expected outcomes, allowing WebPilot to recalibrate its strategy based on new observations. *Controller* plays a crucial role in this process by assessing whether the current observation o_t and the executed action sequences $\boldsymbol{a}$ align with the subtask $\mathcal{T}_i$. It then generates a subtask completeness Comp_t. If the completeness assessment indicates that the subtask is not complete, *Controller* initiates a re-execution of the subtask. Before this re-execution, the Comp_t, along with the associated observation and executed

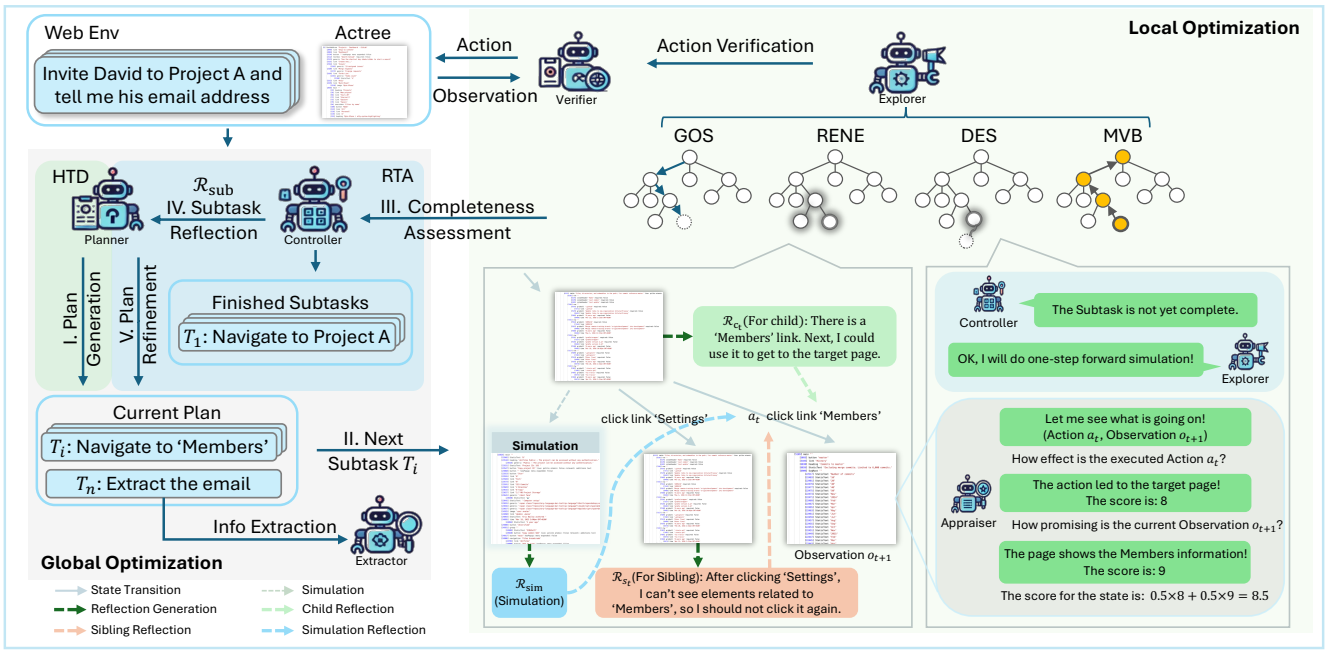

Figure 1: WebPilot, a versatile and autonomous multi-agent system, uses a dual optimization strategy. Global Optimization decomposes tasks and refines plans through reflection, while Local Optimization executes subtasks with customized MCTS.

actions, is used to generate a Strategic Reflection, i.e., subtask reflection $\mathcal{R}_{\text{sub}}$. This reflection guides the repeated execution of the subtask, leveraging the experience from the previous attempt to avoid repeating the same errors. Meanwhile, *Extractor* continuously gathers critical information to support the successful completion of the task.

Local Optimization: MCTS-Enhanced Decision Strategies

The Local Optimization phase is inspired by the human-like adaptability required to navigate and solve complex web tasks, effectively captured by MCTS. For each subtask $\mathcal{T}_i$ and its subtask-specific goals Objective$_i$, which define the expected outcomes or milestones to be achieved within that subtask, *Explorer*, *Verifier*, and *Appraiser* work together to complete the task. *Explorer* identifies optimal actions, *Verifier* ensures these actions are valid and non-redundant, and *Appraiser* evaluates both the immediate effectiveness of an action and its potential to achieve the intended goal, offering continuous feedback for a more nuanced and accurate assessment. Throughout this process, *Controller* assesses whether the subtask is completed and determines if further actions are needed, ensuring alignment with the overall task.

The Local Optimization phase of WebPilot, akin to classical MCTS, follows four key stages: 1) Goal-Oriented Selection leverages the initial intuitions of LLMs to steer WebPilot toward the most promising paths for subtask completion, emulating how humans use prior knowledge to navigate tasks. 2) Reflection-Enhanced Node Expansion integrates feedback after each node expansion, enabling WebPilot to reassess and refine its strategy dynamically, much like human reflection informs decision-making. 3) Dynamic Evaluation and Simulation allows WebPilot to assess current states by analyzing executed actions and simulating potential outcomes, mirroring human foresight in evaluating consequences. 4) Maximal Value Backpropagation prioritizes long-term potential by continuously updating value estimates based on the maximum future rewards. Through this comprehensive Local Optimization phase, WebPilot effectively balances exploration and exploitation, leading to more efficient and informed decision-making in complex tasks.

Goal-Oriented Selection (GOS) directs *Explorer* toward high-value nodes by leveraging the initial intuitions from LLM. Although these intuitions are not specifically tailored to the environment, they provide valuable insights that effectively narrow the action space. These insights stem from the extensive knowledge base of LLM, enabling informed action estimates without explicit web domain training. GOS employs a modified version of the PUCT selection method, inspired by AlphaGo (Silver et al. 2017):

$$U(s,a) = w_{puct} \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}, \tag{1}$$

where w_{puct} represents the exploration bias factor balancing the exploration and exploitation, and $N(s,a)$ represents the total count of conducting the action a in state s.

Unlike classical MCTS, as deployed in RAP (Hao et al. 2023a), which prioritizes unexplored nodes due to the infinite potential assigned by traditional UCT (Kocsis and Szepesvári 2006), GOS refines this approach to better manage vast action spaces. The initial insights of LLM often assign high value to the first visited node based on its broad

knowledge base. In contrast to UCT, which would mandate exploring unvisited nodes even when the current node is nearly optimal, GOS modifies the selection formula by adding a +1 term in the denominator. This adjustment enables GOS to direct the *Explorer* toward the most promising paths, minimizing unnecessary exploration and efficiently navigating complex environments, similar to how humans use prior knowledge in decision-making.

Reflection-Enhanced Node Expansion (RENE) integrates Strategic and Tactical Reflections to refine the strategy of WebPilot, enhancing decision-making through focused exploration and exploitation in vast action spaces.

Specifically, to explore the state space efficiently, WebPilot departs from traditional MCTS by generating and expanding only one action per step. *Explorer* generates actions a_t and corresponding intents $\mathcal{I}_t$ in real time, continuously adjusting to evolving conditions with guidance from reflective feedback. *Verifier* ensures that the action a_t is both valid and unique among sibling nodes. Formally,

$$a_t, \mathcal{I}_t = \textit{Explorer}(o_t, \mathcal{T}_i, H_t, \mathcal{R}_t, \mathcal{C}_{t-1}), \tag{2}$$

where $\mathcal{T}_i$ the subtask, and $H_t = \{a_1, \ldots, a_{t-1}\}$ the action history. If available, the Tactical Reflections—comprising simulation reflection $\mathcal{R}_{\text{sim}_t}$, parent reflection $\mathcal{R}_{p_t}$, and sibling reflection $\mathcal{R}_{s_t}$—along with the Strategic Reflection, which is the subtask reflection $\mathcal{R}_{\text{sub}}$, are incorporated into $\mathcal{R}_t = \{\mathcal{R}_{\text{sim}_t}, \mathcal{R}_{p_t}, \mathcal{R}_{s_t}, \mathcal{R}_{\text{sub}}\}$ and used together with the continuation reason $\mathcal{C}_{t-1}$ to leverage prior exploration. This combined use of available reflections helps narrow the action space and optimize the MCTS process.

Upon executing a_t, the environment returns the resulting state and observation (s_{t+1}, o_{t+1}). *Explorer* then compares the current observation o_{t+1} with the previous one o_t to determine whether the action intent $\mathcal{I}_t$ has been achieved:

$$\text{Eff}(a_t) = \textit{Explorer}(o_{t+1}, o_t, \mathcal{I}_t) \tag{3}$$

Following this evaluation, Child Reflection $\mathcal{R}_{c_t}$ and Sibling Reflection $\mathcal{R}_{s_t}$ are generated:

$$\mathcal{R}_{c_t}, \mathcal{R}_{s_t} = \textit{Explorer}(\text{Eff}(a_t), \mathcal{T}_i, \text{Objective}_i, o_t, H_t), \tag{4}$$

where $\text{Eff}(a_t)$ captures the impact of a_t on the current state, and Objective_i denotes the subtask-specific goals, which define the expected outcomes to be achieved within subtask $\mathcal{T}_i$.

Child Reflection guides the generation of the next action, with $\mathcal{R}_{c_t}$ becoming the parent reflection $\mathcal{R}_{p_{t+1}}$ for the child node, ensuring continuity in decision-making—crucial for complex tasks. Sibling Reflection $\mathcal{R}_{s_t}$ enhances exploration by leveraging insights from previous sibling node explorations, enabling the agent to focus on promising areas and uncover new possibilities when encountering similar scenarios. Together, they help WebPilot narrow the vast action space and integrate feedback from reflections on already executed actions to analyze transitions and assess decisions under incomplete information, improving efficiency and performance in dynamic environments.

Dynamic Evaluation and Simulation (DES) dynamically assesses how generated actions align with evolving task states by leveraging real-time feedback rather than relying on predefined reward structures typical of classical MCTS. This adaptive evaluation ensures that each action remains responsive to the changing environment, guiding the agent toward the evolving goal.

The reward function is crucial in MCTS, but prior methods often rely on direct feedback from the environment (Zhou et al. 2023a), which is impractical for realistic web tasks, or use binary success/failure outcomes or simplistic intermediate states (Koh et al. 2024). Such approaches frequently misjudge the ambiguous and evolving nature of web environments, leading to inaccurate evaluations. Moreover, intermediate steps on the web are challenging to categorize as simply correct or incorrect because their effectiveness in contributing to the final task outcome may not be immediately apparent. Inspired by the A* algorithm (Hart, Nilsson, and Raphael 1968), *Appraiser* evaluates both the effectiveness of the executed action a_t and the potential of the resulting observation o_{t+1} to achieve the intended goal, providing a more nuanced and dynamic assessment. This approach is refined using a granular 0-10 scoring system, which allows for a more precise evaluation of action impact, capturing the subtleties of evolving and uncertain web environments.

This Granular Bifaceted Self-Reward Mechanism evaluates both the effectiveness of the action taken, $S_{\text{eff}}(a_t)$, and the potential of the resulting observation, $S_{\text{fut}}(o_{t+1})$, using a precise 0-10 scoring system. This approach allows for a more nuanced assessment, capturing subtle differences in action quality and future potential, which is crucial for determining whether to proceed with the current strategy.

$$S_{\text{eff}}(a_t), S_{\text{fut}}(o_{t+1}) = \textit{Appraiser}(\text{Eff}(a_t), o_{t+1}, \mathcal{T}_i), \tag{5}$$

where *Apprasier* assesses how well the state aligns with the subtask $\mathcal{T}_i$, and $\text{Eff}(a_t)$ represents the change between the former and current state. The overall reward S_{total} aggregates these scores to represent the value of the current state.

After evaluating the current state, *Controller* determines whether the subtask $\mathcal{T}_i$ is complete,

$$\mathcal{C}_t = \textit{Controller}(\mathcal{T}_i, \{a_1, a_2, \ldots, a_t\}, o_{t+1}), \tag{6}$$

where $\mathcal{C}_t$ is a continuation reason based on executed actions and current observations. If the subtask is complete, the search terminates; otherwise, DES performs a one-step forward simulation guided by RENE, generating Simulation Reflection $\mathcal{R}_{\text{sim}}$ as a shallow trial to inform the next exploration. This simulation helps understand transitions, clarifies ambiguities, and assigns quantitative values to guide more informed decisions, effectively navigating promising paths amid unpredictability and incomplete information.

Maximal Value Backpropagation (MVB) enhances the traditional MCTS backpropagation step by prioritizing the most promising paths. Unlike classical MCTS, which typically averages the values of child nodes and may lead to the exploration of less optimal paths, MVB uses the maximum value from all child nodes, $\max Q(s_{t+1})$, where $Q(s)$ represents the potential value for completing the subtask in state s. For the first visited state, the Q-value is initialized to the score $\mathcal{S}_{\text{total}}$ as evaluated in DES. This approach accumulates values throughout the decision tree, consistently

Method	CMS	Map	Shop.	Red.	Git.	Avg
GPT-3.5						
WebArena (Zhou et al. 2023b)	-	-	-	-	-	8.9
WebPilot (Ours)	**22.0**	**30.3**	**25.1**	**58.5**	**30.0**	**29.1**
GPT-4o						
WebArena (Zhou et al. 2023b)	-	-	-	-	-	13.1
SteP (Sodhi et al. 2024)	24.2	30.3	**36.9**	59.4	31.7	33.5
LM-TS (Koh et al. 2024)	16.5	25.8	28.1	10.5	13.3	19.2
WebPilot (Ours)	**24.7**	**33.9**	**36.9**	**65.1**	**39.4**	**37.2**

Table 2: Performance comparison in WebArena. Values represent SR as percentages. WebPilot achieves SOTA results, showing a 93% relative increase over LM-TS.

targeting strategies with the highest potential for long-term success. By focusing on these high-value paths, WebPilot ensures alignment with the ultimate goal rather than merely advancing to the next immediate step.

Experiment

Experimental Setup

Datasets and Metrics To demonstrate the applicability of WebPilot, we evaluate it on two datasets: WebArena (Zhou et al. 2023b) and MiniWoB++ (Liu et al. 2018). WebArena, with 812 human-annotated tasks, evaluates agent performance on diverse, long-horizon activities in realistic web environments. WebPilot operates as a text-only agent using the accessibility tree (actree) without visual observations—a limitation for future work. WebArena serves as the primary benchmark due to its realism. Additionally, MiniWoB++ evaluates WebPilot on simpler tasks, ranging from button-clicking to form-filling. We use the Success Rate (SR) metric from (Zhou et al. 2023b) for WebArena and follow (Li et al. 2023) to evaluate 43 text-only tasks in MiniWoB++.

Baseline Models We compare WebPilot against SteP (Sodhi et al. 2024) and the concurrent search-based model LM-Tree Search (LM-TS) (Koh et al. 2024). We utilize GPT-3.5 and GPT-4o [1], each configured with a max_tokens limit of 4096 tokens, and a temperature setting of 0.3.

Implementation Details We run WebPilot with parameters optimized for efficiency and performance, limiting the max node count to 10 per subtask and setting exploration bias to 5 to balance exploration and exploitation.

Results on WebArena

Tab. 2 shows that WebPilot consistently outperforms existing methods. In WebArena, WebPilot with GPT-4o demonstrates a remarkable 93% relative increase in SR compared to concurrent tree search-based methods LM-TS, achieving a 37.2% SR. This performance surpasses SteP, which relies on rigid, expert-designed policies tailored to specific states and actions. This substantial improvement highlights the effectiveness of the adaptable and dynamic approach of WebPilot in navigating complex web environments.

[1]GPT-3.5-turbo-0125, GPT-4o-2024-05-13

Greater Flexibility and Adaptability in WebPilot. Compared with SteP, WebPilot demonstrates a significant 7.7% improvement in the Gitlab domain. This advantage arises from the strategic use of high-level demonstrations, which equip the agent with general web domain knowledge rather than confining it to rigid, expert-designed policies specific to certain states and actions, as SteP does. The Gitlab domain, characterized by its diverse and complex tasks, as well as dynamic, multi-step scenarios, highlights the ability of WebPilot to generalize knowledge and adapt strategies in real time. This broader understanding allows WebPilot to more effectively infer and address unseen tasks. Additionally, the exploratory approach of WebPilot, which mirrors human adaptability, facilitates dynamic navigation and problem-solving in unfamiliar scenarios, further enhancing its overall performance in uncertain web environments.

Superior Task Performance in WebPilot Through Strategic Decomposition and Reflective Feedback. WebPilot significantly outperforms LM-TS, particularly in Reddit and GitLab domains, due to two factors. First, LM-TS lacks strategic task decomposition by the *Planner*, making vast state space navigation harder and exploration less efficient. The decomposition of WebPilot highlights its effectiveness. Second, WebPilot employs hierarchical reflections after each node expansion, enabling continuous reassessment and refinement of its strategy.

Enhanced Reasoning and Planning Capabilities Remain Crucial for Improvement. Even with the less powerful GPT-3.5, WebPilot demonstrates a significant improvement over the WebArena baseline, highlighting its effectiveness in leveraging MCTS-inspired strategies to navigate complex environments. However, transitioning from GPT-3.5 to GPT-4o yields substantial gains, particularly in the Shopping, Reddit, and GitLab, with SR increases of 11.8%, 6.6%, and 9.4%, respectively. These improvements are largely driven by the enhanced reasoning and planning capabilities of GPT-4o, which are crucial for tasks requiring precise inference and information retrieval in Shopping, as well as for navigating the more complex and diverse environments of GitLab and Reddit. The advanced planning abilities and the capacity to generalize domain knowledge from limited demonstrations to unseen tasks enable WebPilot to generate more effective plans, better understand the environment, and execute accurate actions. This underscores the importance of addressing the core challenges that current LLMs face in reasoning and planning, suggesting that further enhancements can be achieved with more powerful LLMs.

Results on MiniWoB++

As shown in Tab. 3, WebPilot achieves results comparable to SOTA SteP. The slight edge of SteP comes from using 10 action-level demonstrations, while WebPilot uses only 4 high-level demonstrations, leaving exploration to the agent. The simplicity of MiniWoB++, requiring minimal actions, reduced the need for extensive exploration, limiting the advantage of WebPilot. Despite this, WebPilot performed effectively with far fewer demonstrations than other agents.

Method	SR
RCI (Kim, Baldi, and McAleer 2024)	94.0%
AdaPlanner (Sun et al. 2024)	92.9%
A zero-shot (Li et al. 2023)	94.9%
SteP (Sodhi et al. 2024)	96.0%
WebPilot (Ours)	95.6%

Table 3: Success rate (SR) on MiniWoB++.

Variants	IS	WI
w/o Child Reflection	74%	70%
w/o Sibling Reflection	72%	60%
w/o *Controller*	86%	60%
w/o *Planner*	48%	24%
WebPilot	100%	100%

Table 4: Ablation studies on WebArena.

Ablation Studies

To evaluate the impact of WebPilot components, we categorized tasks into information-seeking (IS) and website interaction (WI), including site navigation and content configuration. IS focuses on extracting information, while WI requires executing complex action sequences. For ablation, we selected 50 IS and 50 WI tasks completed successfully by WebPilot, ensuring components function optimally. Results in Tab. 4 show each component plays a critical role, with WI tasks more adversely affected than IS tasks, highlighting the importance of the design for complex web exploration.

Importance of Child Reflection for Maintaining Coherent Thought Processes. The Child Reflection mechanism is crucial for maintaining a coherent thought process during exploration, aligning with the principles of Chain-of-Thought reasoning (Wei et al. 2022). This reflection ensures that when generating actions for child nodes, the model has access to the parent node behind previous actions and its intended next steps. This continuity enables the model to produce more accurate and contextually relevant actions, preserving the logical flow necessary for complex decision-making. Without Child Reflection, this coherence is disrupted, leading to a significant 30% decline in performance, particularly in WI tasks, where maintaining a consistent thought process is essential for success.

Critical Role of Sibling Reflection in Effective and Diverse Exploration. Sibling Reflection is key to optimizing exploration within MCTS and expanding into diverse, promising areas, particularly for complex tasks. By leveraging insights from previously explored sibling nodes, WebPilot reduces redundancy and focuses on high-potential paths, ensuring valuable solutions are not missed. This mechanism enhances exploration effectiveness, especially for complicated web interaction tasks, as evidenced by WI tasks being more affected, as shown in Tab. 4.

***Controller* is Critical for Subtask Accuracy and High-Level Decision-Making.** *Controller* is crucial for Global Optimization, enabling the reassessment and refinement of plans. After each subtask, *Controller* evaluates its completeness and, in collaboration with *Planner*, refines the overall plan. Without *Controller*, the verification of subtask completion is significantly compromised, leading to a noticeable performance decline, particularly in WI tasks. The greater impact on these tasks, which involve more subtasks, is due to the higher likelihood of incomplete subtasks as their quantity grows. This underscores the vital role of *Controller* in high-level strategic planning and refinement. This finding highlights the importance of trial-and-error, mimicking human problem-solving, where *Controller* serves as a key link between overarching goals and detailed task execution.

***Planner* is Essential for Maximizing WebPilot Performance in Complex Tasks.** The removal of *Planner* significantly degrades performance in complex WI tasks due to task complexity and the high number of subtasks. Task decomposition enhances MCTS effectiveness.

Agent Behavioral Analysis

WebPilot Can Learn Website Functionality by Exploring Unknowns. When switching branches in GitLab, the model first tries a dropdown menu but fails due to too many options, reflecting its limited web knowledge. By exploring further, it adapts and successfully finds the correct branch using the Branch Option in the sidebar, demonstrating its ability to learn and navigate unfamiliar web environments.

WebPilot Can Continuously Adapt Strategies and Actions Based on New Observations. On certain pages, elements like "link Project A" and "statictext Project A" appear multiple times in the actree, referring to the same page. While interacting with the statictext yields no changes, clicking the link achieves the desired outcome. WebPilot observes the error with the statictext, adjusts, and shares this reflection with other nodes, enhancing future decisions and demonstrating its adaptive learning capacity.

WebPilot Can Resolve Ambiguities Through Iterative Refinement. When tasked with checking "Merge request assigned to me", the initial plan involves accessing the merge request and filtering results. However, during the first subtask, WebPilot discovers a more direct path to the goal. Upon successfully completing this subtask, *Controller* can determine that the remaining subtasks have been implicitly accomplished, demonstrating the ability of WebPilot to iteratively refine its strategy to resolve ambiguities.

Conclusion

We introduce WebPilot, a multi-agent system with a dual optimization strategy to enhance the adaptability and effectiveness of autonomous agents in complex web environments. By combining global optimization with a tailored MCTS-based local optimization, WebPilot overcomes the limitations of rigid predefined policies, achieving SOTA results on WebArena. Our approach marks a significant step forward in general autonomous agent capabilities, paving the way for more advanced decision-making in complex environments.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic, A. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1): 1–43.

Chi, Y.; Yang, K.; and Klein, D. 2024. THOUGHTSCULPT: Reasoning with Intermediate Revision and Search. *arXiv preprint arXiv:2404.05966*.

Coulom, R. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, 72–83. Springer.

Daw, N. D.; Niv, Y.; and Dayan, P. 2005. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12): 1704–1711.

Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; and Su, Y. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Fu, Y.; Kim, D.-K.; Kim, J.; Sohn, S.; Logeswaran, L.; Bae, K.; and Lee, H. 2024. Autoguide: Automated generation and selection of state-aware guidelines for large language model agents. *arXiv preprint arXiv:2403.08978*.

Hao, S.; Gu, Y.; Ma, H.; Hong, J.; Wang, Z.; Wang, D.; and Hu, Z. 2023a. Reasoning with Language Model is Planning with World Model. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 8154–8173. Singapore: Association for Computational Linguistics.

Hao, S.; Gu, Y.; Ma, H.; Hong, J. J.; Wang, Z.; Wang, D. Z.; and Hu, Z. 2023b. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Hong, R.; Zhang, H.; Zhao, H.; Yu, D.; and Zhang, C. 2023. Faithful Question Answering with Monte-Carlo Planning. In *Annual Meeting of the Association for Computational Linguistics*.

Jang, Y.; Seo, S.; Lee, J.; and Kim, K.-E. 2021. Monte-carlo planning and learning with language action value estimates. In *International Conference on Learning Representations*.

Kim, G.; Baldi, P.; and McAleer, S. 2024. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, 282–293. Springer.

Koh, J. Y.; McAleer, S.; Fried, D.; and Salakhutdinov, R. 2024. Tree Search for Language Model Agents. *arXiv preprint arXiv:2407.01476*.

Lai, H.; Liu, X.; Iong, I. L.; Yao, S.; Chen, Y.; Shen, P.; Yu, H.; Zhang, H.; Zhang, X.; Dong, Y.; et al. 2024. AutoWebGLM: Bootstrap And Reinforce A Large Language Model-based Web Navigating Agent. *arXiv preprint arXiv:2404.03648*.

Lenz, D.; Kessler, T.; and Knoll, A. 2016. Tactical cooperative planning for autonomous highway driving using Monte-Carlo Tree Search. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, 447–453. IEEE.

Li, T.; Li, G.; Deng, Z.; Wang, B.; and Li, Y. 2023. A Zero-Shot Language Agent for Computer Control with Structured Reflection. *arXiv preprint arXiv:2310.08740*.

Liu, E. Z.; Guu, K.; Pasupat, P.; Shi, T.; and Liang, P. 2018. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *International Conference on Learning Representations (ICLR)*.

Ma, K.; Zhang, H.; Wang, H.; Pan, X.; and Yu, D. 2023. Laser: Llm agent with state-space exploration for web navigation. *arXiv preprint arXiv:2309.08172*.

Pan, J.; Zhang, Y.; Tomlin, N.; Zhou, Y.; Levine, S.; and Suhr, A. 2024. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*.

Prasad, A.; Koller, A.; Hartmann, M.; Clark, P.; Sabharwal, A.; Bansal, M.; and Khot, T. 2023. Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*.

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

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T. P.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the game of Go without human knowledge. *Nature*, 550: 354–359.

Sodhi, P.; Branavan, S. R. K.; Artzi, Y.; and McDonald, R. 2024. SteP: Stacked LLM Policies for Web Actions. arXiv:2310.03720.

Sun, H.; Zhuang, Y.; Kong, L.; Dai, B.; and Zhang, C. 2024. Adaplanner: Adaptive planning from feedback with language models. *Advances in Neural Information Processing Systems*, 36.

Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Tian, Y.; Peng, B.; Song, L.; Jin, L.; Yu, D.; Mi, H.; and Yu, D. 2024. Toward Self-Improvement of LLMs via Imagination, Searching, and Criticizing. *ArXiv*, abs/2404.12253.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Wang, X.; Li, C.; Wang, Z.; Bai, F.; Luo, H.; Zhang, J.; Jojic, N.; Xing, E. P.; and Hu, Z. 2023. PromptAgent: Strategic Planning with Language Models Enables Expert-level Prompt Optimization. *ArXiv*, abs/2310.16427.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xie, Y.; Goyal, A.; Zheng, W.; Kan, M.-Y.; Lillicrap, T. P.; Kawaguchi, K.; and Shieh, M. 2024. Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning. *arXiv preprint arXiv:2405.00451*.

Yang, Z.; Li, L.; Lin, K.; Wang, J.; Lin, C.-C.; Liu, Z.; and Wang, L. 2023. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1): 1.

Zhang, D.; Huang, X.; Zhou, D.; Li, Y.; and Ouyang, W. 2024. Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B. *ArXiv*, abs/2406.07394.

Zhao, Z.; Lee, W. S.; and Hsu, D. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

Zheng, L.; Wang, R.; Wang, X.; and An, B. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*.

Zhou, A.; Yan, K.; Shlapentokh-Rothman, M.; Wang, H.; and Wang, Y.-X. 2023a. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2023b. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.