

AgentCollab: A Self-Evaluation-Driven Collaboration Paradigm for Efficient LLM Agents

Wenbo Gao^{1,2*}, Renxi Liu², Xian Wang², Fang Guo², Shuai Yang², Xi Chen²,
Hui-Ling Zhen², Hanting Chen², Weizhe Lin^{2†}, Xiaosong Li², Yaoyuan Wang²

¹The Hong Kong Polytechnic University, ²Huawei
wenbo.gao@connect.polyu.hk linweizhe1@huawei.com

Abstract

Autonomous agents powered by large language models (LLMs) perform complex tasks through long-horizon reasoning and tool interaction, where a fundamental trade-off arises between execution efficiency and reasoning robustness. Models at different capability–cost levels offer complementary advantages: lower-cost models enable fast execution but may struggle on difficult reasoning segments, while stronger models provide more robust reasoning at higher computational cost. We present *AgentCollab*, a self-driven collaborative inference framework that dynamically coordinates models with different reasoning capacities during agent execution. Instead of relying on external routing modules, the framework uses the agent’s own self-reflection signal to determine whether the current reasoning trajectory is making meaningful progress, and escalates control to a stronger reasoning tier only when necessary. To further stabilize long-horizon execution, we introduce a difficulty-aware cumulative escalation strategy that allocates additional reasoning budget based on recent failure signals. In our experiments, we instantiate this framework using a two-level small–large model setting. Experiments on diverse multi-step agent benchmarks show that AgentCollab consistently improves the accuracy–efficiency Pareto frontier of LLM agents.

1 Introduction

Autonomous agents powered by large language models (LLMs) (Zhang et al., 2026b; Li et al., 2024) are increasingly deployed to tackle complex tasks such as web-based information seeking (Zheng et al., 2025; Qiao et al., 2025; Li et al., 2025), advanced mathematical reasoning (Kapoor et al., 2026; Yan et al., 2025), and long-form writing (Xia et al., 2025). Unlike traditional single-turn

question answering systems, these agents formulate plans, interact with external tools (e.g., web searching, document reading, and text composition), and iteratively refine their reasoning based on observed outcomes (Shi et al., 2025). This iterative Think–Act–Observe loop enables agents to solve complex problems but also introduces substantial latency. An early mistake can propagate through subsequent steps and significantly degrade the final outcome (Zhang et al., 2026a; Kapoor et al., 2026). Allocating stronger reasoning capacity can mitigate such errors and improve reasoning quality. However, each reasoning step then becomes more expensive, and the accumulated latency grows rapidly in multi-turn scenarios. A fundamental dilemma therefore emerges between execution speed and reasoning quality.

Existing acceleration techniques mainly focus on improving efficiency for single-turn inference (Xia et al., 2024; Egashira et al., 2024) or short-horizon orchestration (Ong et al., 2025; Chen et al., 2024; Wang et al., 2025). In contrast, the efficiency objective arises at a different level in autonomous agents due to long-horizon reasoning and interaction. Performance must be evaluated over the entire reasoning trajectory rather than individual inference steps. When an early mistake requires several additional cycles to correct, the time saved in a single step quickly becomes negligible compared with the extra turns spent on recovery. Correctness at a few critical steps often dominates overall efficiency, suggesting that optimization strategies should operate at the trajectory level instead of uniformly accelerating every step.

Recent work has begun to explore model routing as a way to balance capability and efficiency. Early studies mainly focus on *single-turn* model selection, where a router chooses between cheaper and more capable models for a given query. Systems such as RouteLLM by Ong et al. (2025) and FrugalGPT by Chen et al. (2024) employ lightweight

*Work done during internship at Huawei.

†Corresponding author.

routers to evaluate outputs from lower-cost models and invoke stronger models when necessary. These approaches primarily optimize monetary or token-level cost at the single-response level, which limits their relevance to long-horizon agents, where the central objective is end-to-end inference speed rather than per-query cost alone.

Later work by Zhang et al. (2026a) and Kapoor et al. (2026) extends routing to *multi-step* agents by learning policies that select a model at each step of the reasoning process. To reduce inference cost, Zhang et al. (2026a) proposes allocating stronger models to steps predicted to be difficult but solvable. To improve reasoning quality, Kapoor et al. (2026) leverages progress reward models to assess intermediate reasoning steps. While these methods move beyond single-turn routing, their main focus remains cost reduction or quality improvement. By contrast, our goal is to improve end-to-end reasoning speed in long-horizon agent execution. Under this objective, routing introduces additional challenges: routing decisions must be made repeatedly during execution, extra supervision is required to learn the routing policy, and frequent switching may interrupt computational reuse such as prefill states. The resulting overhead can offset the latency gains of lower-cost models, particularly in long reasoning trajectories. More fundamentally, for long-horizon agents the key challenge is not merely *which model to call at a step*, but how control over the reasoning trajectory should be allocated and transferred across different levels of reasoning capacity.

These observations suggest a different perspective on efficiency. Instead of introducing external routing models or learned policies, the agent itself can already reflect on its reasoning process. That capability can serve as a natural signal for coordinating different levels of reasoning capacity. From this perspective, model collaboration can be driven internally by the reasoning process rather than controlled by an external router.

Following this intuition, we introduce a **self-driven collaboration paradigm** for multi-turn agents. In this paradigm, the reasoning process itself determines when stronger reasoning capacity is required. A lower-cost model tier performs most reasoning steps and continuously evaluates whether the current trajectory is making meaningful progress. When stagnation is detected, control is temporarily escalated to a stronger reasoning tier that resolves the difficult segment of the tra-

jectory. Once progress resumes, control can return to a more efficient tier and the interaction loop continues. This process requires no auxiliary routing model and no additional policy training, as the decision signal is generated by the model’s own self-reflection.

Such collaboration naturally extends beyond a fixed pair of models. The same principle can coordinate a hierarchy of models with different capability–cost trade-offs, allowing reasoning capacity to be gradually increased only when necessary. This forms a scalable paradigm for allocating reasoning resources across long trajectories, where inexpensive tiers handle routine steps while stronger tiers intervene at critical moments.

Based on this paradigm, we further introduce a cumulative escalation strategy that adjusts the duration of higher-capacity intervention according to historical stagnation signals. Repeated failures lead to longer takeovers, enabling deeper correction on difficult trajectory segments while reducing oscillatory switching.

Our contributions are threefold:

- We introduce a **self-driven collaboration paradigm** for multi-turn LLM agents, in which the agent’s own trajectory-level reflection signals determine how control is allocated across models with different capability–cost trade-offs, without external routers or learned routing policies.
- We propose a **difficulty-aware cumulative budget allocation strategy** that regulates higher-capacity intervention budget based on consecutive stagnation signals, enabling progressively stronger intervention under persistent reasoning difficulty.
- We conduct experiments on diverse multi-turn agent benchmarks including web-based research, mathematical reasoning, and long-form writing. The results demonstrate a strong latency–quality trade-off, consistently improving reasoning efficiency while maintaining high performance.

2 AgentCollab

This section introduces *AgentCollab*, a collaborative efficient inference paradigm that dynamically allocates computational resources among language models of different sizes during multi-turn agent

execution. The framework addresses a fundamental limitation of fixed-model agent systems. Small models offer fast inference but often struggle in complex reasoning segments, while large models provide stronger reasoning but incur substantially higher single-step latency when used throughout the entire trajectory. Relying on a single model therefore leads to an inherent trade-off between efficiency and robustness in long-horizon workflows.

As illustrated in Figure 1, AgentCollab introduces a model escalation mechanism that enables adaptive collaboration between small and large models during inference. The small model serves as the default controller for routine reasoning steps, while the large model is invoked only when the trajectory encounters difficult segments. Building upon this mechanism, we further develop a *difficulty-aware budget allocation strategy* that regulates the intervention strength of the large model. Instead of assigning a fixed intervention budget, the system adjusts the duration of large-model takeover based on recent failure signals along the trajectory. As shown in Figure 2, repeated stagnation progressively increases the escalation budget, enabling stronger corrective reasoning when the agent repeatedly fails to make progress.

2.1 Model Escalation

Multi-step agent reasoning consists of sequential “Think” operations interleaved with tool interaction, during which certain trajectory segments may become locally trapped and lead to oscillation or redundant exploration. Persistent use of the large model M_L reduces this risk at the expense of substantial computational cost, whereas exclusive reliance on the small model M_S improves efficiency but weakens robustness in difficult regions.

To balance these effects, we design a **self-evaluation-driven escalation mechanism** that dynamically coordinates LLMs with different capabilities. As the early-stage reasoning strongly influences the subsequent trajectory and therefore benefits from higher-capacity inference (Zhang et al., 2026a; openPangu Team, 2025), the session begins with a short warm-up phase executed by strong model M_L . The global task framing, intermediate objective decomposition, and high-level strategy planning are performed in this phase. After initialization, control is transferred to small model M_S , which carries out the majority of later reasoning steps and tool calls in a low-latency routine.

To determine when to escalate, at each step t ,

the model produces not only a reasoning output but also a structured progress assessment, which is generated in parallel with the normal reasoning output. This reflection takes the form of a canonical block containing a *rationale* field and a binary indicator *value*. The rationale summarizes whether the latest reasoning steps advance the trajectory toward the final objective, while the indicator encodes this judgment as a binary variable $\text{value}_t \in \{\text{TRUE}, \text{FALSE}\}$, where $\text{value}_t = \text{TRUE}$ corresponds to meaningful progress and $\text{value}_t = \text{FALSE}$ corresponds to stagnation. The signal is generated by the model itself based solely on the current reasoning context, and does not rely on any training process or auxiliary routing model.

When $\text{value}_t = \text{TRUE}$, the controller maintains execution under the small model, allowing well-conditioned subproblems to be solved with minimal cost. When $\text{value}_t = \text{FALSE}$, the trajectory is interpreted as entering a difficult region under the current capacity. The large model then takes over the control and executes for a bounded number of reasoning steps, during which it re-examines the accumulated context and restructures intermediate states if necessary. When the predefined intervention budget is exhausted or progress is made, execution subsequently returns to M_S . This bounded escalation–de-escalation process forms a closed feedback loop in which self-evaluation governs model routing. High-capacity computation is therefore concentrated on genuinely difficult segments while overall inference cost remains controlled.

2.2 Difficulty-aware Budget Allocation Strategy

The basic escalation mechanism determines when control should be transferred to the large model, yet it does not regulate the strength of intervention once escalation occurs. A small budget bound may lead to a weak intervention, failing to resolve persistent stagnation. However, an excessively large budget quota risks overusing escalation, increasing the latency instead.

To address this limitation, a **difficulty-aware cumulative budget allocation strategy** that controls intervention depth at each escalation according to the recent self-evaluation history is proposed as shown in Figure 2. We first define the escalation level l_t as the number of consecutive failure signals up to step t , which is recursively updated as

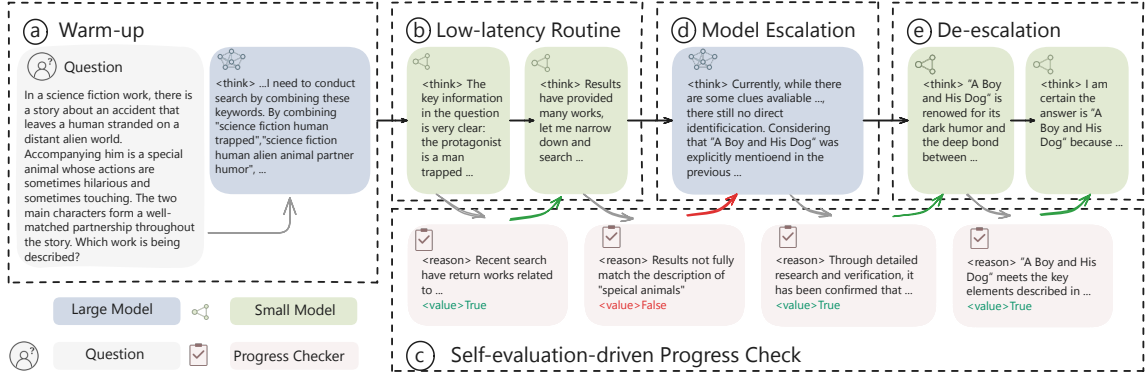


Figure 1: **Overview of the proposed AgentCollab framework.** (a) The agent system first invokes the large model to warm up the interaction and plan the overall strategy. (b) After initialization, the small model performs low-latency routine reasoning steps and (c) conducts self-evaluation on progress checks to determine whether escalation is required. (d) When stagnation is detected, control is temporarily transferred to a larger model within a predefined budget to resolve difficult reasoning segments. (e) Once the critical step is completed, control returns to the small model to continue the interaction.

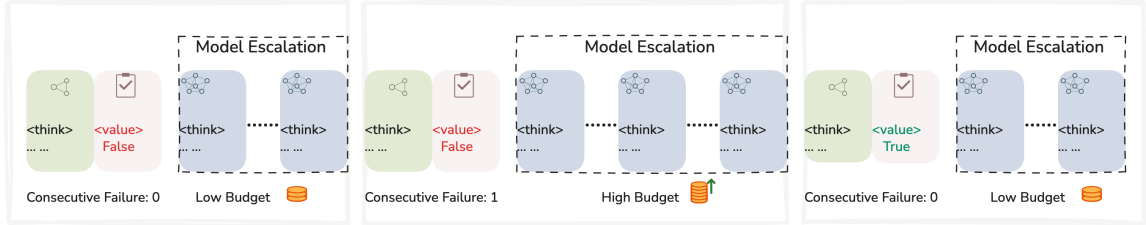


Figure 2: **Budget Allocation Strategy.** As consecutive failures accumulate, the framework temporarily increases the budget for the larger model, allowing stronger models to provide additional support.

$$l_t = \begin{cases} l_{t-1} + 1, & \text{if } \text{value}_t = \text{FALSE}, \\ 0, & \text{if } \text{value}_t = \text{TRUE}. \end{cases} \quad (1)$$

The level therefore captures the accumulated difficulty reflected by trajectory-level self-evaluation.

Let B^0 denote the large-model intervention budget when $l = 0$. The actual intervention horizon at escalation level l is defined as a bounded growth function:

$$B^l = f(B^0, l) = \min\{B^{\max}, B^0 + kl\}, \quad (2)$$

where $k \geq 0$ controls the growth rate and $B^{\max}$ is the maximum intervention budget. When $k = 0$, the strategy reduces to static model switching. When repeated stagnation increases l , the large model receives a longer but still bounded intervention budget, enabling stronger correction without unbounded latency growth.

Although $f(\cdot)$ may take other bounded forms, such as sigmoid or saturating schedules, we adopt the linear bounded formulation because it is simple, interpretable, and sufficient for controlling the intervention strength in our experiments.

3 Experiments and Discussion

3.1 Setup

To evaluate the generality and effectiveness of the AgentCollab mechanism, we integrate it into two open-source agent frameworks, DDV2 (open-Pangu Team, 2025) and WebSailor (Li et al., 2025), which provide pretrained models of different scales (7B/38B and 3B/7B/32B respectively). We evaluate the framework on diverse multi-turn tasks, including deep research (BrowseComp_zh (Zhou et al., 2025)), challenging mathematical reasoning (HLE with math subjects (Center for AI Safety et al., 2026)), and long-form generation (Writing-Bench (Wu et al., 2025)). The three benchmarks contain 289, 866, and 1000 questions, respectively. Appendix results on GAIA (Mialon et al., 2024) and XBench-DeepSearch (Chen et al., 2025) further cover additional deep-research benchmarks. For fair comparison, all experiments use a maximum interaction budget of 40 iterations. The budget allocation strategy follows the linear formulation described in Section 2.2, with $B^0 = 2$ and $k = 2$. All inference is conducted locally using

Agent	Method	BrowseComp_zh			HLE-math			WritingBench		
		Acc. (%)	#Steps	Speedup	Acc. (%)	#Steps	Speedup	Score	#Steps	Speedup
DDV2	Small	18.3	21.75	1.54 ×	8.0	35.55	3.38 ×	4.4	17.25	3.20 ×
	Large	34.6	17.29	1.00×	23.3	28.41	1.00×	5.1	9.10	1.00×
	Random	22.3	20.56	1.12×	9.2	32.82	1.33×	4.2	13.10	1.29×
	RouteLLM	28.2	22.40	1.09×	15.2	33.42	1.62×	4.7	10.27	2.21×
	FrugalGPT	27.2	20.20	1.29×	18.2	29.23	1.97×	4.5	12.22	1.94×
	TRiM	29.6	21.38	1.18×	20.4	30.18	2.05×	4.6	12.86	1.71×
	Miroflow	31.2	20.74	1.23×	18.7	31.64	1.84×	4.8	11.14	2.02×
	WeDAS	30.8	19.96	1.28×	19.1	30.92	1.91×	4.8	10.92	2.08×
	Ours	<u>33.9</u>	19.82	<u>1.36</u> ×	<u>21.1</u>	32.03	<u>2.31</u> ×	<u>5.0</u>	9.59	<u>2.43</u> ×
WebSailor	Small	14.2	10.38	2.03 ×	11.3	13.02	2.48 ×	-	-	-
	Large	25.5	7.79	1.00×	14.0	7.09	1.00×	-	-	-
	Random	14.9	10.48	1.20×	9.2	14.02	1.03×	-	-	-
	RouteLLM	20.0	9.60	1.32×	11.4	9.34	1.21×	-	-	-
	FrugalGPT	19.7	10.50	1.28×	11.9	9.81	1.10×	-	-	-
	TRiM	18.8	10.12	1.22×	13.6	8.92	1.18×	-	-	-
	Miroflow	21.4	9.18	1.35×	12.5	9.08	1.16×	-	-	-
	WeDAS	21.0	9.06	1.39×	12.7	8.96	1.19×	-	-	-
	Ours	<u>22.5</u>	8.49	<u>1.50</u> ×	13.3	8.57	<u>1.29</u> ×	-	-	-

Table 1: **Main results on BrowseComp_zh, HLE-math, and WritingBench.** Speedup is measured relative to the large-model baseline in each agent framework. AgentCollab consistently approaches large-model performance while preserving clear efficiency gains. Results for WritingBench are not reported for WebSailor because it does not support long-form writing tasks. Best results are shown in **bold**, and second-best results are underlined.

vLLM-Ascend (v0.9.1) on Ascend 910B3 NPUs with four workers.

For baselines, we compare AgentCollab with small-model-only and large-model-only agents, as well as several model-switching methods. *Random* selects between the small and large models uniformly at random at each reasoning step. RouteLLM (Ong et al., 2025) and FrugalGPT (Chen et al., 2024) were originally designed for cost-efficient single-turn routing, and we adapt them to the multi-turn setting by making routing decisions at each reasoning step. To cover multi-turn evaluation settings, we further include TRiM (Kapoor et al., 2026), Miroflow (Su et al., 2026), and WeDAS (Yu et al., 2026). All switching baselines operate over the same small-large model pairs used by AgentCollab.

Three metrics are reported: accuracy score, speedup, and the average number of reasoning iterations per question. Accuracy is evaluated by GPT-4o (OpenAI, 2023), while score is evaluated by the model provided by the benchmark (Wu et al., 2025). Speedup measures the end-to-end latency improvement relative to the large-model baseline and is defined as the ratio of the total latency of the large-model agent to that of the evaluated method. The iteration count reflects the reasoning cycles required to reach the final answer, with small models often requiring more rounds to recover from earlier mistakes.

3.2 Main Results

Table 1 reports the results on three representative multi-turn agent benchmarks covering web research (BrowseComp_zh), mathematical reasoning (HLE-math), and long-form generation (WritingBench). Overall, AgentCollab substantially improves small-model agents while preserving clear efficiency advantages over large-model agents.

As shown in Table 1 and Figure 3, on BrowseComp_zh, AgentCollab improves performance over the small-model agents for both DDV2 (18.3→33.9) and WebSailor (14.2→22.5), while remaining close to the large-model agents (34.6 and 25.5, respectively). Similar trends are observed on HLE-math, where collaboration substantially improves reasoning accuracy (e.g., 8.0% → 21.1%

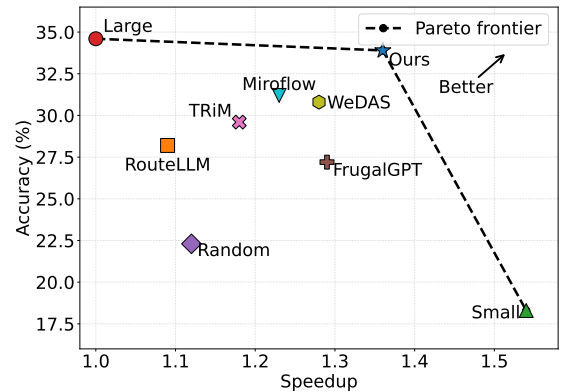


Figure 3: Pareto frontier of DDV2 on BrowseComp_zh.

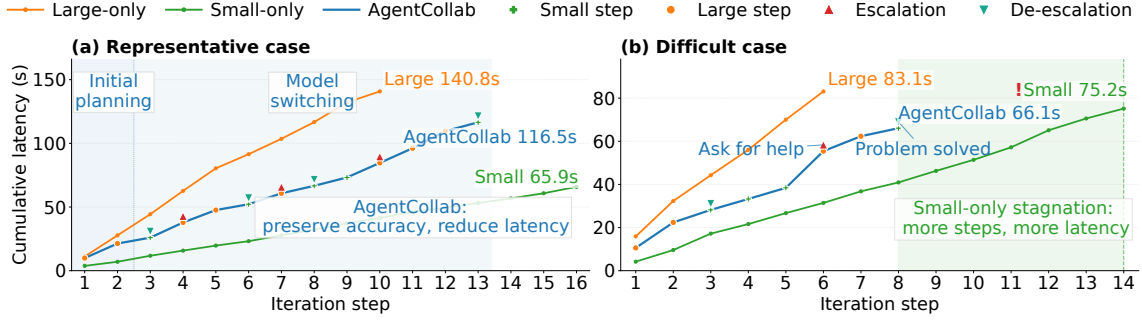


Figure 4: **Illustrative execution trajectories.** Cumulative latency is shown across iteration steps for Large-only, Small-only, and AgentCollab. AgentCollab markers indicate the model used at each step, and triangles mark escalation or de-escalation events. In the representative case (a), AgentCollab reduces latency relative to Large-only execution by invoking the Large model only when necessary. In the difficult case (b), Large-only execution resolves the task within fewer steps, whereas Small-only execution becomes trapped in an unproductive reasoning trajectory and accumulates higher latency. AgentCollab escalates when needed and returns to the Small model after recovery.

for DDV2) while still preserving clear speedup over the large-model baseline. On WritingBench, the collaborative system also improves generation quality (4.4→5.0) and remains close to the large-model performance (5.1).

These results highlight a key challenge in multi-turn agents: although small models provide faster inference per step, they often require more interaction rounds to complete the task. AgentCollab mitigates this issue by invoking stronger models only when necessary, often leading to shorter reasoning trajectories while maintaining high efficiency.

Across all benchmarks, the collaborative approach achieves a favorable *efficiency–quality frontier*: it substantially outperforms the small-model agents and approaches the performance of large-model agents, while preserving substantial efficiency gains over the large-model baseline. This demonstrates that the proposed self-reflection-driven escalation mechanism can effectively concentrate large-model computation on more difficult segments of the trajectory.

Compared with other model-switching baselines, AgentCollab achieves the strongest overall accuracy–efficiency trade-off. Random switching can achieve nontrivial speedup through frequent use of the small model, but the lack of trajectory awareness leads to weak reasoning performance. RouteLLM (Ong et al., 2025) and FrugalGPT (Chen et al., 2024) improve over random switching by adapting single-turn routing mechanisms to step-level decisions, while TRiM (Kapoor et al., 2026), Miroflow (Su et al., 2026), and WeDAS (Yu et al., 2026) provide stronger multi-turn baselines. Nevertheless, these methods remain

less effective than AgentCollab at allocating high-capacity computation along the full reasoning trajectory.

3.3 Ablation Study

To analyze the effectiveness of the proposed collaboration mechanism, we compare several configurations of small and large model allocation on the DDV2 agent system. The study examines the following aspects: the role assignment between planner and executor, the escalation strategy used in AgentCollab including the sensitivity of the budget growth parameter k .

Method	Acc. (%)	Speedup
Large-model only	34.6	1.00×
Small-model only	18.3	1.54×
<i>Role Assignment</i>		
w/ Large Planner	24.6	1.39×
w/ Large Executor	27.3	1.24×
<i>Escalation Strategy</i>		
AgentCollab (Static)	32.5	1.32×
AgentCollab (Dynamic)	33.9	1.36×

Table 2: **Ablation study on DDV2.** We examine the effect of planner/executor role assignment and escalation strategy. Accuracy and speedup are reported.

Impact of Planner and Executor Roles. The DDV2 agent explicitly separates two roles: a *planner* and an *executor* (information seeker). The planner produces high-level reasoning plans, while the executor performs information retrieval and action execution. This structure allows us to examine how allocating the large model to different roles affects overall performance.

We enable the large model in these two roles separately. As the results shown in Table 2, using the

large model in the execution stage while keeping the planner as a small model yields higher accuracy than assigning the large model to the planner (27.3 vs. 24.6), although the speedup decreases slightly (from $1.39\times$ to $1.24\times$). This result suggests that the execution stage is the more error-sensitive component in multi-turn agents: it directly interacts with the environment, determines the quality of retrieved evidence, and more strongly influences the subsequent reasoning states. As a result, allocating stronger models to execution is more effective for improving the speed–quality frontier.

Impact of Budget Allocation Strategy. We further compare two escalation mechanisms in AgentCollab: a static allocation strategy and the proposed dynamic allocation strategy. Both configurations substantially outperform the small-model baseline, while the dynamic strategy achieves a better quality–efficiency trade-off than the static strategy. It improves accuracy from 32.5 to 33.9 and slightly increases speedup from $1.32\times$ to $1.36\times$, indicating that allocating larger intervention budgets after repeated stagnation is more effective than using a fixed escalation budget.

This improvement is also reflected in the switching behavior. A key motivation for the dynamic strategy is to reduce oscillatory switching, which can weaken large-model prefill caching by interrupting consecutive reasoning segments. We quantify this effect using the switching ratio defined in Equation 3 in the appendix. The dynamic strategy lowers the switching ratio from 49.64 to 45.07, suggesting fewer alternations between the small and large models. This supports our claim that difficulty-aware budget growth encourages more contiguous large-model intervention on difficult trajectory segments, thereby stabilizing execution and improving cache reuse.

To study budget allocation, we vary the hyperparameter k from 0 to 3, where a larger k yields more aggressive growth of the large-model intervention budget and $k = 0$ reduces to the static strategy. A small budget may fail to resolve persistent stagnation, whereas an excessively large budget may overuse the large model and increase latency without proportional accuracy gains. The dynamic strategy balances these effects by increasing the intervention budget only after repeated stagnation, while keeping the total intervention bounded by the maximum budget. As shown in Figure 5, $k = 2$ achieves the best quality–efficiency trade-off. This suggests that a moderate cumulative growth rate

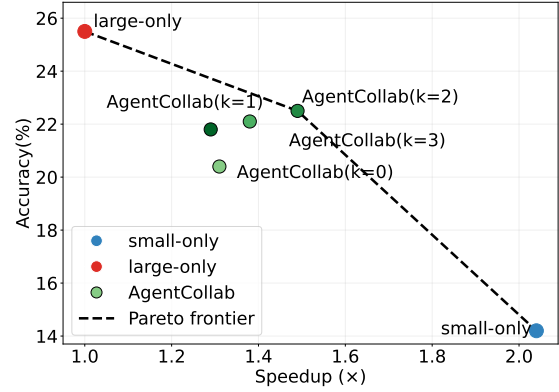


Figure 5: Pareto frontier of AgentCollab within WebSailor on BrowseComp_zh under different values of k .

best balances corrective strength and efficiency.

3.4 Analysis

Trajectory-Level Execution Analysis. To better understand how AgentCollab behaves during multi-step execution, we visualize cumulative latency and model-escalation trajectories at the case level in Figure 4. In the representative case, AgentCollab reduces latency relative to Large-only execution by invoking the Large model only when necessary. The trajectory therefore grows more slowly than the Large-only baseline while still allowing stronger computation to intervene at selected steps.

The difficult case highlights why the fastest model at each step does not necessarily yield the most efficient trajectory. Small-only execution can become trapped in an unproductive reasoning path, requiring more iterations and accumulating higher total latency. AgentCollab avoids this inefficient trajectory by escalating when progress stalls and returning to the Small model after recovery. These examples suggest that end-to-end efficiency depends on both per-step cost and trajectory quality: a faster step is beneficial only when the reasoning process remains productive.

Impact of Agent Framework. We observe that AgentCollab performs less satisfactorily on WebSailor than on DDV2. One possible reason is the difference in instruction-following ability across the underlying models. WebSailor is built on an earlier Qwen2.5-based setup (Qwen Team, 2024). An important observation is that WebSailor is more prone to false-positive progress judgments: even when the trajectory has not made meaningful progress, it is more likely to predict that progress has been made. Empirically, the false progress-check rate is around 3% in DDV2 (openPangu

Agent	1-step Recovery	2-step Recovery
DDV2	96.8%	99.9%
WebSailor	95.2%	99.7%

Table 3: **Recovery rate after false-positive progress judgments.** Measured by manual inspection, most misjudgments are corrected within one step, and nearly all are corrected within two steps.

Agent	Frequency	Acc. (%)	#Steps	Speedup
DDV2	Every step	33.9	19.82	1.36×
	Every 3 steps	32.7	20.46	1.40×
	Every 5 steps	31.8	21.17	1.43×
WebSailor	Every step	22.5	8.49	1.50×
	Every 3 steps	20.7	8.93	1.55×
	Every 5 steps	19.4	9.48	1.59×

Table 4: **Ablation on progress-check frequency on BrowseComp_zh.** Speedup is measured relative to the large-model baseline in each agent framework.

Team, 2025), compared with 7% in WebSailor (Li et al., 2025). In addition, DDV2 deploys a multi-process search agent, which provides more robust research results. Weaker instruction following and less robust search make the self-evaluation signal less reliable and, in turn, weaken the effectiveness of model escalation. However, as shown in Table 3, most progress-check misjudgments are corrected within the following one or two rounds, which limits their negative impact on the overall trajectory.

Progress Check Frequency. Although the progress signal is generated in parallel with the normal reasoning output rather than through an additional inference call, its overhead is worth examining. We therefore evaluate whether reducing the frequency of self-evaluation can improve efficiency. As shown in Table 4, less frequent progress checks only slightly improve the speedup, but this gain is accompanied by more reasoning steps and substantially lower accuracy in both DDV2 and WebSailor. The modest speedup mainly comes from increased small-model usage, as delayed checking postpones large-model intervention and leaves more iteration steps to the small model. However, this also allows unproductive or incorrect trajectories to persist longer before correction, offsetting the small saving from less frequent checking. Therefore, reducing progress-check frequency does not provide a clear end-to-end benefit, and we use step-wise progress checking as the default setting.

Heterogeneous Model Pairing. We further evaluate collaboration between models that differ not only in size but also in model family. As shown in

Model Pair	Acc. (%)	#Steps	Speedup
<i>Baseline same-family pairing</i>			
Pangu-7B + Pangu-38B	33.9	19.82	1.36×
Qwen-7B + Qwen-32B	32.4	17.64	1.49×
<i>Cross-family pairing</i>			
Pangu-7B + Qwen-32B	31.9	18.72	1.41×
Qwen-7B + Pangu-38B	32.6	18.05	1.44×

Table 5: **Heterogeneous model-pair ablation under the DDV2 framework on BrowseComp_zh.** The first block reports same-family baseline pairings, while the second block evaluates cross-family collaborations. Cross-family pairings retain competitive accuracy and speedup, suggesting that AgentCollab remains feasible beyond same-family model collaboration.

Table 5, same-family pairings show a slight advantage, suggesting that shared training backgrounds may lead to more consistent reasoning styles, tool-use behaviors, and progress-check interpretations. Such cross-model alignment remains challenging in heterogeneous model collaboration (Chen et al., 2026; Hu et al., 2025). Nevertheless, the gap is modest: Pangu-7B + Qwen-32B achieves 31.9 accuracy with 1.41× speedup, and Qwen-7B + Pangu-38B achieves 32.6 accuracy with 1.44× speedup. These results suggest that AgentCollab is not limited to the same-family size scaling and can operate across models with different pretraining backgrounds.

Failure analysis. To better understand the remaining limitations of AgentCollab, we compare successful and failed cases in terms of reasoning steps, large-token ratio, and total latency. Here, the large-token ratio denotes the proportion of tokens generated by the large model and serves as a proxy for reliance on the large model. Figure 6 in the appendix shows that failed cases generally involve longer trajectories, greater reliance on the large model, and higher end-to-end latency.

4 Conclusion

This paper presents *AgentCollab*, a collaborative inference framework that dynamically coordinates language models of different sizes for long-horizon agent reasoning. Instead of relying on external routing modules or fixed model allocation, the framework adopts a self-evaluation-driven escalation mechanism, allowing the model itself to determine whether the current reasoning trajectory is making meaningful progress and whether stronger intervention is needed. To improve multi-step execution, we incorporate a difficulty-aware budget allocation strategy that adaptively regulates large-

model intervention under persistent stagnation. Experimental results show that AgentCollab achieves a favorable trade-off between reasoning quality and execution latency, improving the Pareto frontier across diverse agent benchmarks. More broadly, the proposed framework points to a promising direction for efficient agent systems, where heterogeneous language models collaborate through internal signals rather than external orchestration.

Limitations

The main efficiency evaluation is conducted with open-source LLMs deployed through local inference services, where latency, batching behavior, hardware allocation, and decoding configurations can be controlled consistently across methods. We also provide additional experiments with closed-source API models, suggesting that the proposed collaboration mechanism remains applicable in API-based settings. However, API latency is affected by server-side scheduling and network conditions, making it difficult to fairly compare end-to-end speedup with locally deployed models. We therefore report accuracy and reasoning steps for API-based experiments, while leaving a more controlled analysis of API latency and monetary cost to future work.

References

- Center for AI Safety, Scale AI, and HLE Contributors Consortium. 2026. [A benchmark of expert-level academic questions to assess AI capabilities](#). *Nature*, 649:1139–1146.
- Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, and 14 others. 2025. [xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations](#). *arXiv preprint arXiv:2506.13651*.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2024. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.
- Shilian Chen, Jie Zhou, Qin Chen, Wen Wu, Xin Li, Qi Feng, and Liang He. 2026. Can heterogeneous language models be fused? *arXiv preprint arXiv:2604.01674*.
- Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. 2024. Exploiting llm quantization. *Advances in Neural Information Processing Systems*, 37:41709–41732.
- Jinwu Hu, Yufeng Wang, Shuhai Zhang, Kai Zhou, Guohao Chen, Yu Hu, Bin Xiao, and Minghui Tan. 2025. Efficient dynamic ensembling for multiple llm experts. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 8095–8103.
- Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. 2026. Trim: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks. *arXiv preprint arXiv:2601.10245*.
- Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Websailor: Navigating superhuman reasoning for web agent. *arXiv preprint arXiv:2507.02592*.
- Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicnearth*, 1(1):9.
- Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. [Gaia: a benchmark for general ai assistants](#). In *International Conference on Learning Representations*, volume 2024, pages 9025–9049.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Kadous, and Ion Stoica. 2025. [Routellm: Learning to route llms from preference data](#). In *International Conference on Learning Representations*, volume 2025, pages 34433–34448.
- OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- openPangu Team. 2025. [openpangu-deepdiver-v2 technical report](#). Technical report, Huawei. Accessed: 2025-12-03.
- Zile Qiao, Guoxin Chen, Xuanzhong Chen, Donglei Yu, Wenbiao Yin, Xinyu Wang, Zhen Zhang, Baixuan Li, Huifeng Yin, Kuan Li, Rui Min, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Webresearcher: Unleashing unbounded reasoning capability in long-horizon agents. *arXiv preprint arXiv:2509.13309*.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- Wenxuan Shi, Haochen Tan, Chuqiao Kuang, Xiaoguang Li, Xiaozhe Ren, Chen Zhang, Hanting Chen, Yasheng Wang, Lifeng Shang, Fisher Yu, and Yunhe Wang. 2025. Pangu deepdiver: Adaptive search intensity scaling via open-web reinforcement learning. *arXiv preprint arXiv:2505.24332*.

- Shiqian Su, Sen Xing, Xuan Dong, Muyan Zhong, Bin Wang, Xizhou Zhu, Yuntao Chen, Wenhai Wang, Yue Deng, Pengxiang Zhu, Ziyuan Liu, Tiantong Li, Jiaheng Yu, Zhe Chen, Lidong Bing, and Jifeng Dai. 2026. Miroflow: Towards high-performance and robust open-source agent framework for general deep research tasks. *arXiv preprint arXiv:2602.22808*.
- Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025. Mixllm: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922.
- Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. 2025. Writingbench: A comprehensive benchmark for generative writing. *arXiv preprint arXiv:2503.05244*.
- Haotian Xia, Hao Peng, Yunjia Qi, Bin Xu, Juanzi Li, Hou Lei, and Xiaozhi Wang. 2025. [Storywriter: A multi-agent framework for long story generation](#). In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM '25*, page 6559–6563, New York, NY, USA. Association for Computing Machinery.
- Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhi-fang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7655–7671.
- Yibo Yan, Shen Wang, Jiahao Huo, Philip S Yu, Xuming Hu, and Qingsong Wen. 2025. Mathagent: Leveraging a mixture-of-math-agent framework for real-world multimodal mathematical error detection. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 69–82.
- Zixuan Yu, Zhenheng Tang, Tongliang Liu, Chengqi Zhang, Xiaowen Chu, and Bo Han. 2026. Rethinking deep research from the perspective of web content distribution matching. *arXiv preprint arXiv:2603.07241*.
- Caiqi Zhang, Menglin Xia, Xuchao Zhang, Daniel Madrigal, Ankur Mallick, Samuel Kessler, Victor Ruehle, and Saravan Rajmohan. 2026a. Budget-aware agentic routing via boundary-guided training. *arXiv preprint arXiv:2602.21227*.
- Kaiyan Zhang, Kai Tian, Runze Liu, Sihang Zeng, Xuekai Zhu, Guoli Jia, Yuchen Fan, Xingtai Lv, Yuxin Zuo, Che Jiang, Yuru wang, Jianyu Wang, Ermo Hua, Xinwei Long, Junqi Gao, Youbang Sun, Zhiyuan Ma, Ganqu Cui, Ning Ding, and 2 others. 2026b. [MARTI: A framework for multi-agent LLM systems reinforced training and inference](#). In *The Fourteenth International Conference on Learning Representations*.
- Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 414–431.
- Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. 2025. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*.

A Additional Benchmark

To further examine whether the observed accuracy–efficiency trade-off generalizes beyond the main benchmarks, we report additional results on GAIA and XBenCh in Table 6. These benchmarks cover complementary multi-step reasoning and tool-use scenarios. The results show a consistent trend with the main experiments: AgentCollab improves over the small-model baseline while retaining clear efficiency advantages over the large-model baseline, suggesting that the proposed collaboration mechanism is not specific to a single benchmark or agent framework.

B Closed-source API

We also evaluate AgentCollab with closed-source API models, as reported in Table 7. This setting tests whether the proposed collaboration mechanism remains effective when the underlying models are accessed only through black-box inference APIs. The results show that AgentCollab can still provide a favorable balance between task performance and inference cost, indicating that the method is compatible with both open-source local deployment and practical API-based agent systems.

C Definition of the Switching Ratio

The switching ratio used to quantify oscillatory switching is defined as:

$$R_{\text{switch}} = \frac{\sum_{i=1}^N \sum_{t=2}^{T_i} \mathbb{I}(m_{i,t} \neq m_{i,t-1})}{\sum_{i=1}^N (T_i - 1)}. \quad (3)$$

where N is the number of trajectories, T_i is the number of steps in trajectory i , $m_{i,t} \in \{M_S, M_L\}$ denotes the selected model at step t , and $\mathbb{I}(\cdot)$ is the indicator function.

D Case Study

To further illustrate how AgentCollab corrects stalled reasoning trajectories through model escalation, we present a representative case in Table 8, comparing small-model-only, large-model-only, and AgentCollab under the same task. The small-model-only setting becomes trapped in a difficult subproblem and gradually drifts toward an incorrect search direction, whereas the large-model-only setting is able to maintain broad and accurate exploration throughout the trajectory. AgentCollab initially encounters a similar difficulty, but detects the lack of meaningful progress through

self-evaluation and temporarily transfers control to the large model. After the difficult segment is resolved, control returns to the small model, which resumes efficient reasoning on the corrected trajectory. This example shows that the proposed collaboration mechanism can recover from local stagnation while preserving much of the efficiency advantage of the small model.

E Difficulty-Split Analysis of Progress-Check Frequency

To further examine how progress-check frequency affects tasks with different reasoning complexity, we split the BrowseComp_zh instances into simple and hard subsets according to the number of reasoning steps required by the default step-wise progress-check setting. Specifically, tasks with fewer than 10 steps for DDV2 and fewer than 5 steps for WebSailor are treated as simple tasks, while the remaining tasks are treated as hard tasks. This analysis complements the main progress-check frequency ablation by showing whether the observed accuracy–efficiency trade-off differs across task difficulty levels.

As shown in Table 10, reducing the progress-check frequency has a smaller effect on simple tasks, where the reasoning horizon is short and delayed escalation is less likely to affect the final answer. By contrast, hard tasks obtain larger speedup because more reasoning steps are executed by the small model before escalation is triggered, but this also leads to larger accuracy degradation. These results are consistent with the main ablation in Table 4: less frequent checking can slightly improve efficiency, but delayed intervention increases the risk that the small model remains on an unproductive trajectory for longer. We therefore use step-wise progress checking as the default setting.

F Difficulty-Aware Speedup Analysis

To better understand how the efficiency gain varies across task complexity, we analyze the per-instance speedup ratio with respect to task difficulty. Following a simple operational proxy, we use the execution latency of the large model as an estimate of task difficulty: instances that require longer large-model execution are treated as more difficult.

Figure 7 shows that the speedup of both AgentCollab and the small-only baseline increases with task difficulty. This trend suggests that harder tasks provide more room for reducing latency relative to

Agent	Method	GAIA			xbench-DeepSearch		
		Acc. (%)	#Steps	Speedup	Acc. (%)	#Steps	Speedup
DDV2	Small	35.9	24.18	1.68 ×	39.0	27.32	1.82 ×
	Large	54.2	18.52	1.00×	53.0	20.74	1.00×
	Random	38.6	23.04	1.19×	41.8	26.15	1.17×
	RouteLLM	45.1	24.58	1.16×	45.6	27.40	1.14×
	FrugalGPT	44.3	22.36	1.33×	46.4	24.92	1.31×
	TRiM	45.8	23.21	1.24×	47.2	25.84	1.23×
	Miroflow	49.6	21.84	1.31×	49.8	24.31	1.34×
	WeDAS	48.9	21.27	1.37×	49.1	23.76	1.40×
	Ours	<u>52.3</u>	21.03	<u>1.45</u> ×	<u>51.7</u>	23.18	<u>1.52</u> ×
WebSailor	Small	37.9	12.04	2.16 ×	34.3	13.47	2.29 ×
	Large	53.2	8.31	1.00×	53.3	8.86	1.00×
	Random	40.7	12.28	1.22×	37.5	13.86	1.15×
	RouteLLM	45.6	10.76	1.34×	43.2	11.72	1.28×
	FrugalGPT	44.9	11.34	1.29×	42.6	12.36	1.21×
	TRiM	44.1	11.06	1.25×	41.8	12.08	1.19×
	Miroflow	48.7	10.18	1.38×	47.4	10.94	1.36×
	WeDAS	48.2	9.92	1.43×	46.8	10.62	1.41×
	Ours	<u>51.0</u>	9.36	<u>1.57</u> ×	<u>50.6</u>	10.03	<u>1.55</u> ×

Table 6: **Additional results on GAIA and xbench-DeepSearch.** Speedup is measured relative to the large-model baseline in each agent framework. AgentCollab maintains strong accuracy-efficiency trade-offs on additional deep-search benchmarks.

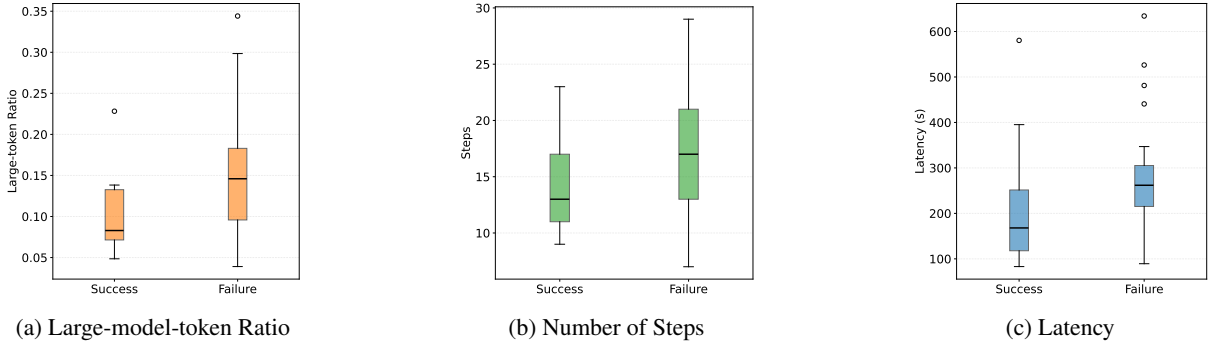


Figure 6: **Failure analysis.** Box plots comparing successful and failed cases in terms of (a) large-token ratio (the proportion of tokens generated by the Large model), (b) number of reasoning steps, and (c) latency.

Model Pair	Acc. (%)	#Steps
GPT-5-mini + GPT-5	32.8	20.35
Gemini-3-Flash + Gemini-3-Pro	31.6	20.91

Table 7: **Closed-source API model-pair results on BrowseComp_zh under the DDV2 framework.** We report accuracy and reasoning steps only, since API latency is affected by server-side scheduling and network conditions.

the large-only setting. The small-only baseline exhibits a steeper increase, which is expected because it always executes the smaller model and therefore achieves greater latency reduction as the number of steps increases. By contrast, AgentCollab selectively invokes the large model when self-evaluation indicates insufficient progress. This design moderates the absolute speedup gain, but it preserves

the corrective capability of the large model on challenging trajectory segments.

These results support the intended trade-off of AgentCollab. Rather than maximizing speed alone, the framework maintains substantial efficiency gains as task difficulty grows, while reserving stronger computation for cases in which purely small-model execution is more likely to fail.

G Effect of Model Size Pairing.

To further examine how the choice of model sizes affects collaboration performance, we compare different small-large model pairings on WebSailor over BrowseComp_zh, as shown in Table 9. Here, configurations such as 3B + 7B denote collaboration between two models of different sizes. For fair comparison, speedup is computed relative to the

Small Model	AgentCollab	Large Model
Let me analyze this task: The task requires searching for a historically significant document jointly signed by multiple national representatives, with a focus on the English copy stored in a Chinese museum. I should begin with a web search.	Let me analyze this task: I need to search for a multi-nation jointly signed historical document, stored in a Chinese museum, in English. Possible candidates include the UN Charter, Treaty of Versailles, etc. I need both Chinese and English keywords.	Let me analyze this task: Find an English document jointly signed by multiple national representatives with major historical significance, and stored in a Chinese museum. I should focus on international treaties, UN documents, and museum collections in China.
Analysis: All three systems begin with high-quality reasoning from the large model. <i>The Small Model mode also starts with large model planning to compare them in a more fair configuration.</i>		
.....		
Now I have obtained detailed information on the Cairo Declaration... Let me search for other similar important documents.	I still have not found a multi-nation signed English document stored in a Chinese museum. Let me try searching for the “UN Conference Memorial Album” from the Museum of Overseas Chinese History. [Progress check: no significant progress made. Escalate.]	I found important information: UN Charter drafting documents (Chongqing Library), “UN Conference Memorial Album” (Museum of Overseas Chinese History), and other multi-nation historical materials.
Analysis: The left system enters a local loop with limited search breadth; the AgentCollab system encounters stagnation and triggers model escalation; the right system continues high-quality knowledge expansion.		
.....		
From the second round of search results, I did not find other documents that clearly meet the requirements... Let me search the UN Charter or other international treaties.	Great! I found the “UN Conference Memorial Album”: stored in the Museum of Overseas Chinese History, involving 50 nations, containing bilingual UN Charter texts, fully meeting the task requirements. [Restore to small model.]	Summary of findings: English UN Charter, UN Conference Memorial Album, “Records of Liberated Areas of China,” and other relevant documents.
Analysis: This is where collaborative reasoning shows its clear advantage: the left system fails to find the key document; the middle system, after escalation, finds the exact same crucial evidence as the large-model-only system.		
.....		
Now I have enough information to complete the task: Cairo Declaration, UN Charter (museum collection unclear).	I found three classes of documents: the UN Conference Memorial Album; UN documents stored in the National Library of China; and international convention files stored through UN depository library systems (including English versions).	Let me reorganize all findings: English UN Charter, UN Conference Memorial Album, Records of Liberated Areas of China, and UN Charter drafting documents — ready to finish the task.
Analysis: The right system maintains maximal coverage at high cost; the middle system reaches nearly the same breadth and accuracy through collaboration; the left system misses the core target and yields the lowest final quality.		

Table 8: **Comparison of key THINK steps across three reasoning modes:** small-model-only, AgentCollab, and large-model-only.

Configuration	Acc. (%)	#Step	Speedup
3B	9.7	9.44	1.37×
3B + 7B	12.5	11.40	1.08×
7B	14.2	10.38	–
3B + 32B	12.8	9.08	1.40×
7B + 32B	22.5	8.49	1.50×
32B	25.5	7.79	–

Table 9: **WebSailor on BrowseComp_zh with different model size configurations.** Configurations such as 3B + 7B denote collaboration between two models of different sizes. Speedup is computed against 7B in the upper block and against 32B in the lower block.

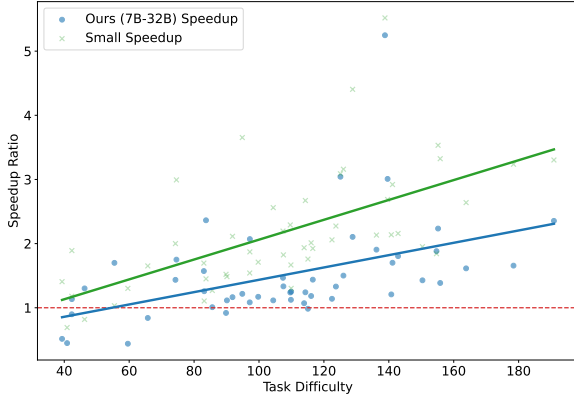


Figure 7: **Speedup ratio as a function of task difficulty.** Task difficulty is measured by the total latency of the Large (32B) baseline for each task (x-axis), and the y-axis reports the corresponding speedup ratio. Blue circles denote the speedup of Ours (7B–32B), while green crosses denote the speedup of the Small-only model (7B). Solid lines show linear trend fits. The red dashed line at $y = 1$ indicates parity with the Large baseline. The proposed AgentCollab maintains a clear positive trend under a selective large-model allocation strategy.

7B baseline in the upper block and relative to the 32B baseline in the lower block.

As shown in Table 9, collaboration becomes more effective when the capability gap between the paired models is larger. In particular, 7B + 32B substantially improves over 7B alone and approaches the performance of 32B, while requiring fewer reasoning steps and achieving 1.50× speedup. By contrast, the gain from 3B + 7B is relatively limited, suggesting that the benefit of collaboration depends not only on model combination itself, but also on whether the stronger model can provide sufficient corrective capacity over the weaker one.

H Pseudo-code of AgentCollab

Algorithm 1 presents the full inference procedure of AgentCollab with static budget allocation strategy. The process consists of an initial planning stage with the large model, followed by a collaborative reasoning stage in which the small model handles routine progress and the large model is invoked only when self-evaluation indicates stagnation. During each escalation episode, the number of consecutive large-model steps is bounded by a predefined budget to avoid excessive computation. The procedure terminates once a final answer is produced from the accumulated context.

I Progress-check Prompt

For the progress check described in Section 2, we prepend the following instruction to the model’s self-reflection step. This prompt requires the model to (i) summarize the newly observed actions and outcomes since the previous THINK step, (ii) assess whether these updates move the trajectory meaningfully closer to the final objective, and (iii) return a binary judgment in a strictly constrained output format.

Agent	Task Type	Acc. (3-step)	Speedup	Acc. (5-step)	Speedup
DDV2	Simple	33.6 (-0.3)	1.37×	33.3 (-0.6)	1.39×
	Hard	33.0 (-0.9)	1.42×	32.4 (-1.5)	1.47×
WebSailor	Simple	22.1 (-0.4)	1.52×	21.8 (-0.7)	1.54×
	Hard	21.1 (-1.4)	1.58×	20.1 (-2.4)	1.63×

Table 10: **Difficulty-split analysis of progress-check frequency on BrowseComp_zh.** Accuracy drops relative to step-wise progress checking are shown in parentheses. Simple tasks achieve limited speedup with minimal accuracy degradation, whereas hard tasks gain more efficiency but suffer larger performance drops.

Prompt for Progress Check

In this THINK step, I will first summarize what has changed since the previous THINK step,
then analyze whether these changes constitute significant progress toward the final objective.

Summary of new actions and results since the previous THINK step:

-

Impact of these changes on progress toward the objective:

-

Now I will make an explicit binary judgment about progress.

IMPORTANT: In the PROGRESS block below, I MUST strictly follow this structural format:

===PROGRESS===

<reason> one or two sentences explaining whether there is significant progress and why
</reason>

<value>TRUE or FALSE only</value>

===END_PROGRESS===

and finish the generation immediately without any further content. Ignore the original requirements for the think tool.

Now I write the actual PROGRESS block following this structural format:

===PROGRESS===

<reason>

Algorithm 1: AgentCollab with Self-Evaluation-Driven Escalation

Input: User query Q , Large model M_L , Small model M_S , Initial large-model budget K_L , Maximum consecutive large-model steps per escalation B_L

Output: Final answer A

// Phase 0: Initialization

$ctx \leftarrow \text{init_context}(Q)$;

$mode \leftarrow \text{LARGE}$;

$large_steps_used \leftarrow 0$;

// Phase 1: Initial Planning with Large Model

while $large_steps_used < K_L$ **do**

$think_out \leftarrow M_L.\text{THINK}(ctx)$;

$ctx \leftarrow ctx \cup \{think_out\}$;

$prog \leftarrow M_L.\text{PROGRESS_CHECK}(ctx)$;

$large_steps_used \leftarrow large_steps_used + 1$;

if $prog.value = \text{TRUE}$ **then**

if $is_final_answer(ctx)$ **then**

$A \leftarrow \text{extract_answer}(ctx)$;

return A ;

$mode \leftarrow \text{SMALL}$;

// Phase 2: Collaborative Reasoning with Self-Evaluation-Driven Escalation

while not $is_final_answer(ctx)$ **do**

if $mode = \text{SMALL}$ **then**

$think_out \leftarrow M_S.\text{THINK_AND_TOOLS}(ctx)$;

$ctx \leftarrow ctx \cup \{think_out\}$;

$prog \leftarrow M_S.\text{PROGRESS_CHECK}(ctx)$;

if $prog.value = \text{TRUE}$ **then**

continue;

else

$mode \leftarrow \text{LARGE}$;

$large_steps_used \leftarrow 0$;

continue;

else if $mode = \text{LARGE}$ **then**

$think_out \leftarrow M_L.\text{THINK_AND_TOOLS}(ctx)$;

$ctx \leftarrow ctx \cup \{think_out\}$;

$prog \leftarrow M_L.\text{PROGRESS_CHECK}(ctx)$;

$large_steps_used \leftarrow large_steps_used + 1$;

if $prog.value = \text{TRUE}$ **then**

if $is_final_answer(ctx)$ **then**

$A \leftarrow \text{extract_answer}(ctx)$;

return A ;

$mode \leftarrow \text{SMALL}$;

else if $large_steps_used \geq B_L$ **then**

$mode \leftarrow \text{SMALL}$;

// Phase 3: Finalization

$A \leftarrow \text{extract_answer}(ctx)$;

return A ;
