

000 RETHINKING THE VALUE OF MULTI-AGENT WORK- 001 FLOW: A STRONG SINGLE AGENT BASELINE 002

003 **Anonymous authors**
004

005 Paper under double-blind review
006

007 ABSTRACT 008

009 Recent advances in LLM-based multi-agent systems (MAS) show that workflows
010 composed of multiple LLM agents with distinct roles, tools, and communication
011 patterns can outperform single-LLM baselines on complex tasks. However, most
012 frameworks are homogeneous, where all agents share the same base LLM and differ
013 only in prompts, tools, and positions in the workflow. This raises the question
014 of whether such workflows can be simulated by a single agent through multi-
015 turn conversations. We investigate this across [seven benchmarks spanning coding, math-
016 ematics, general question answering, domain-specific reasoning, and real-
017 world planning and tool use](#). Our results show that a single agent can reach the
018 performance of homogeneous workflows with an efficiency advantage from KV
019 cache reuse, and can even match the performance of an automatically optimized
020 heterogeneous workflow. Building on this finding, we propose **OneFlow**, an al-
021 gorithm that automatically tailors workflows for single-agent execution, reducing
022 inference costs compared to existing automatic multi-agent design frameworks
023 without trading off accuracy. These results position the single-LLM implemen-
024 tation of multi-agent workflows as a strong baseline for MAS research. We also
025 note that single-LLM methods cannot capture heterogeneous workflows due to the
026 lack of KV cache sharing across different LLMs, highlighting future opportunities
027 in developing *truly* heterogeneous multi-agent systems.
028

029 1 INTRODUCTION 030

031 Recent advances in large language models (LLMs) have sparked significant interest in multi-agent
032 systems (MAS), where multiple LLM agents collaborate through predefined workflows to tackle
033 complex tasks. These systems typically consist of specialized LLM agents, each defined by distinct
034 system prompts and tools, that communicate according to specific patterns to achieve superior per-
035 formance compared to single-LLM approaches (Zhuge et al., 2024; Liu et al., 2024; Zhang et al.,
036 2025e). Current research has demonstrated the effectiveness of such multi-agent workflows across
037 diverse domains, from mathematical reasoning to code generation and tool usage (Hu et al., 2025;
038 Zhang et al., 2025b; Wang et al., 2025b).

039 However, a critical observation about existing MAS reveals a fundamental characteristic that has
040 been largely overlooked: **most current multi-agent systems are homogeneous** (Ye et al., 2025a;
041 Zhang et al., 2025a). Within a given MAS, all agents rely on the same base LLM, differentiated only
042 by their system prompts, tools, and positions in the workflow. This homogeneity raises a compelling
043 question: [if all agents share the same underlying model and work collaboratively to solve a task, can a single agent simulate the multi-agent workflow effectively through multi-turn conversations?](#)

044 It is well recognized that task decomposition is critical for solving complex problems, which directly
045 motivates the design of multi-agent systems. Given a multi-agent workflow that already decomposes
046 a task, what happens if a single agent executes the workflow end-to-end? Specifically, because
047 homogeneous agents possess identical reasoning capabilities and differ only in their specialized
048 instructions, a single agent should be capable of role-playing these agents sequentially, thereby
049 exploiting the workflow’s task decomposition. Moreover, a single agent can reuse a shared KV cache
050 across agent interactions, retaining context without additional prefill cost and potentially offering
051 efficiency gains and greater consistency than maintaining separate model instances for each agent.
052

To investigate this hypothesis, we conduct comprehensive experiments across [seven benchmarks spanning coding, mathematics, general question answering, domain-specific reasoning, and real-world planning and tool use](#). Our results reveal that *a single agent using multi-turn conversations with KV cache can indeed simulate tailored workflows with performance comparable to traditional homogeneous multi-agent setups, while reducing cost*. This finding challenges the conventional assumption that multiple separate agent instances are necessary for effective tailored reasoning.

Building on this insight, we introduce **OneFlow**, an algorithm for automatically designing tailored workflows that optimize both performance and computational efficiency for single-agent execution. Based on recent work on automatic workflow design (Zhang et al., 2025e; b; Nie et al., 2025; Hu et al., 2025), OneFlow employs a dual-LLM designer architecture to discover streamlined workflows with longer, more comprehensive system prompts for individual agents and fewer total agents in the system. This approach achieves similar performance to existing methods while significantly reducing inference cost.

We further extend our analysis to heterogeneous multi-agent workflows, where agents use different base LLMs (Ye et al., 2025a; Zhang et al., 2025a; Wang et al., 2024a). Exhaustively exploring combinations of base models can be very expensive (Ye et al., 2025a). [In a pilot experiment using AFlow \(Zhang et al., 2025e\) to automatically design heterogeneous workflows via Monte Carlo Tree Search, we find that our single-LLM baseline can match the performance of one such automatically discovered heterogeneous alternative with less computational cost](#). Importantly, single-LLM approaches have inherent limitations: they cannot simulate *truly* heterogeneous workflows due to the inability to share KV caches across different models. [This limitation suggests that developing effective heterogeneous agentic workflows, where the benefits of model diversity outweigh coordination costs, remains a promising and necessary direction for advancing LLM multi-agent system research](#). Our contributions can be summarized as follows:

- [Empirically validate that single-agent execution can effectively simulate **homogeneous** multi-agent workflows with comparable performance on collaborative tasks](#).
- Propose **OneFlow**, an algorithm for automatic workflow design that generates streamlined multi-agent architectures with improved computational efficiency and suitability for single-agent execution.
- [Show in a pilot study that a single-LLM implementation can match the performance of one automatically discovered heterogeneous workflow; more importantly, realizing the full benefits of heterogeneity remains an important open direction for agentic LLM systems](#).

2 PRELIMINARY

LLM-based Multi-Agent Workflows. Multi-agent workflows represent a paradigm where multiple LLM agents collaborate through structured communication patterns to solve complex tasks. To illustrate this concept, consider the session-based query recommendation example shown in Figure 1. Given a customer’s shopping history and current query, the task requires understanding session context, analyzing product relationships, and generating relevant recommendations, a multi-faceted problem that benefits from specialized processing stages, as demonstrated in the middle panel of Figure 1. Importantly, these workflows are implemented as executable Python code (right panel of Figure 1) that specifies both the agents and their interaction logic. This code-based representation enables complex control flows including sequential execution, conditional branching, iterative refinement, and collaborative deliberation patterns.

Formal Definition. We now provide a formal characterization of LLM-based multi-agent workflows. Let \mathcal{M} denote the set of available base LLMs. An LLM-based multi-agent workflow W is defined as a directed graph $G = (N, E)$ where:

- $N = \{n_1, n_2, \dots, n_{|N|}\}$ represents the set of LLM agents. Each agent n_i is parameterized as $n_i = (b_i, p_i, \tau_i)$, where: $b_i \in \mathcal{M}$ is the base LLM (e.g., Claude 3.5 Haiku, Gemini 2.5 Flash), p_i is the system prompt that defines the agent’s role and capabilities, τ_i is the available tool set (e.g., sandboxed Python interpreter, web search). Agents are typically specialized for specific subtasks, with common roles including LLM reviewers, LLM ensemblers, and output formatters.

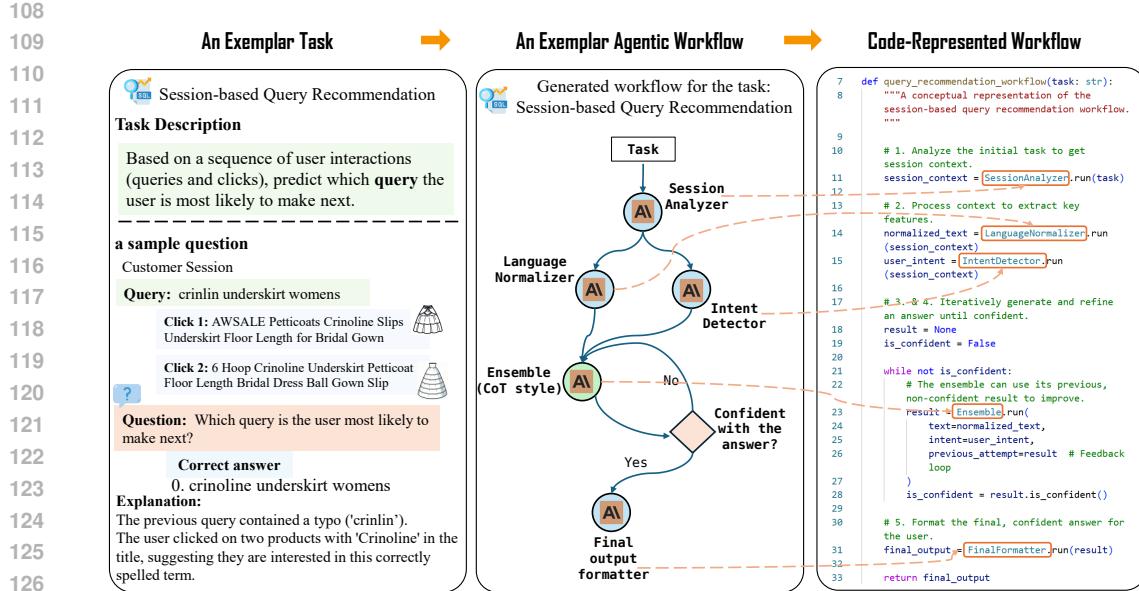


Figure 1: Sample question–answer pair from a session-based query recommendation task (left), an exemplar agentic workflow to solve it (middle), and its code representation (right). The workflow demonstrates how multiple LLM agents can collaborate to process complex shopping queries through sequential and conditional execution patterns.

- $E \subseteq N \times N$ encodes the inter-agent communication structure and control flow. Each edge may include routing conditions and message transformations implemented in Python, enabling sophisticated orchestration patterns such as sequential processing, conditional branching, and iterative loops.

Homogeneous vs. Heterogeneous Workflows. A critical distinction in multi-agent workflows concerns the diversity of underlying base models. We define $\mathcal{B}(W) = \{b_i \mid n_i \in N\}$ as the set of base LLMs used by workflow W . Based on this, workflows can be categorized as:

- **Homogeneous workflows:** $|\mathcal{B}(W)| = 1$, where all agents share the same base LLM and differ only in their system prompts, tools, and positions within the workflow structure.
- **Heterogeneous workflows:** $|\mathcal{B}(W)| > 1$, where agents utilize different base LLMs, potentially leveraging diverse model capabilities and specializations.

Design Complexity of heterogeneous workflows. The choice of which base model to assign to each agent (the mapping $i \mapsto b_i$) represents a non-trivial design decision that is often determined empirically. While heterogeneous workflows offer greater design flexibility by combining models with complementary strengths, they also significantly expand the design space to include model selection alongside prompt engineering, tool assignment, and routing logic (models \times prompts \times tools \times routing). Consequently, many existing automatic workflow design systems default to homogeneous configurations for practical reasons.

KV Cache. The distinction between **heterogeneous** and **homogeneous** workflows is fundamental to our analysis, as it determines whether a workflow can be efficiently simulated by a single agent instance through multi-turn conversations with shared KV cache. In transformer-based LLMs, the key-value (KV) cache is a crucial optimization technique that stores the computed key and value matrices from attention layers for previously processed tokens. Without KV caching, the model would redundantly recompute these attention states for all previous tokens when generating each new token, leading to quadratic computational complexity. By caching these intermediate states, the model achieves significant speedup during autoregressive generation. In **homogeneous** multi-agent workflows, where all agents share the same base LLM, there exists substantial contextual overlap between agent interactions, such as shared task descriptions, intermediate reasoning steps,

162 and common knowledge bases. This overlap enables efficient KV cache sharing across different
 163 agent roles within a single LLM instance, potentially offering both computational efficiency gains
 164 and improved consistency compared to maintaining separate model instances for each agent.
 165

166 3 METHODOLOGY

168 3.1 SINGLE AGENT CAN BE AS STRONG AS MULTI-AGENT FRAMEWORK.

170 We formalize when a single agent can implement a homogeneous multi-agent workflow without
 171 loss of expressivity. Recall the formalization in the Preliminary section: a workflow W is a directed
 172 graph $G = (N, E)$ with agents $n_i = (b_i, p_i, \tau_i)$ and routing logic on edges. Let W be homogeneous
 173 with $|\mathcal{B}(W)| = 1$ and base LLM b . Executing W on input x produces a transcript

$$174 \quad H_T = (h_0, m_1, r_1, h_1, \dots, m_T, r_T, h_T),$$

175 where at step t the workflow selects agent index i_t and tool action a_t according to a policy $\pi_W(i_t, a_t \mid$
 176 $h_{t-1})$ induced by E , queries the same base model b with system prompt p_{i_t} and context h_{t-1} to
 177 obtain model message m_t , optionally executes a tool $u_t \in \tau_{i_t}$ to obtain result r_t , and updates the
 178 history h_t .
 179

180 Consider a single-LLM simulator that maintains one conversation state and, at each step t , sets the
 181 system message to p_{i_t} , appends the same visible context h_{t-1} and tool outputs, and decodes from
 182 the same base model b with identical decoding parameters.

183 **Proposition 1 (Simulation of homogeneous workflows).** Suppose (i) tool side-effects are deter-
 184 ministic given inputs, (ii) the routing policy π_W depends only on the visible history h_{t-1} and tool
 185 outputs, and (iii) decoding uses deterministic rules (e.g., greedy) or shared randomness. Then the
 186 single-LLM simulator induces the same distribution over transcripts as executing W with separate
 187 agent instances:
 188

$$H_T^{\text{single}} \stackrel{d}{=} H_T^{\text{multi}}.$$

189 *Proof sketch.* Both procedures query the same conditional distribution $b(\cdot \mid p_{i_t}, h_{t-1})$ at the same
 190 sequence of states; induction on t yields equality in distribution.

191 **Cost with KV cache.** Let L_t be the tokenized length of the prefix visible at step t and ΔL_t the
 192 number of new tokens appended between steps $t-1$ and t . With separate agent instances (no cache
 193 sharing), overlapping prefixes are re-encoded, yielding cost

$$194 \quad C_{\text{multi}} \propto \sum_{t=1}^T L_t^{(i_t)} + \text{gen}_t.$$

197 The single-LLM simulator reuses the same KV cache across t , so

$$199 \quad C_{\text{single}} \propto \sum_{t=1}^T \Delta L_t + \text{gen}_t \leq C_{\text{multi}},$$

202 with equality only when agent contexts are disjoint. Thus, for homogeneous workflows with sub-
 203 stantial contextual overlap, single-LLM execution is asymptotically no worse and often cheaper,
 204 while preserving behavior under the conditions above.

205 3.2 SINGLE AGENT IMPLEMENTATION OF MULTI-AGENT WORKFLOW.

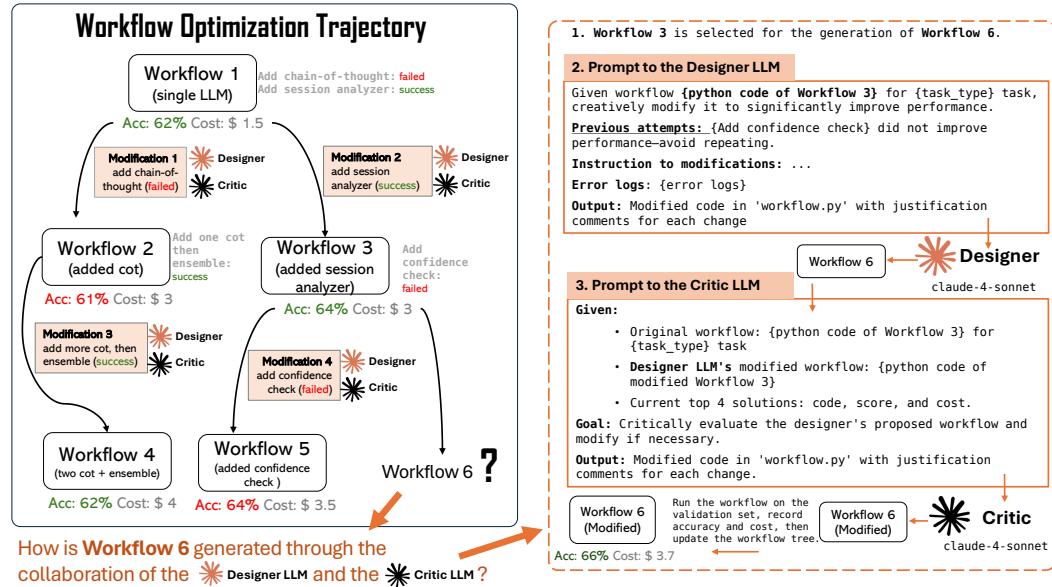
207 We provide a concrete single-LLM simulator for any homogeneous workflow $W = (N, E)$ with
 208 base LLM b . Let $n_i = (b, p_i, \tau_i)$. The simulator maintains a single chat history h_t and follows the
 209 routing policy encoded by E .
 210

- 211 1. **Initialization.** Set $h_0 = \text{wrap}(x)$ with task x and global instructions. Insert role delimiters
 212 to isolate subsequent agent turns.
- 213 2. **Agent step t .** Using the routing logic in E , select index i_t and required tool action a_t based
 214 on h_{t-1} . **We append the system message p_{i_t} to the end of the conversation history h_{t-1} (effectively
 215 treating the ‘system message’ as a user message), keeping all previous context.** Then, query b to obtain
 $m_t \sim b(\cdot \mid p_{i_t}, h_{t-1})$ with fixed decoding parameters.

216 3. **Tool execution.** If a_t invokes a tool $u \in \tau_{i_t}$, execute u to get result r_t and append it to the
217 history.
218 4. **State update.** Set $h_t = \text{append}(h_{t-1}, (i_t, m_t, r_t))$ and advance according to edge conditions
219 in E . Repeat until termination.
220

221 Because every step calls the same base model b , the simulator reuses the KV cache across t (see
222 Preliminary), so the prefill cost scales with incremental growth ΔL_t rather than the full prefix L_t .
223 To control context growth and mitigate interference, one may insert compaction operators (e.g.,
224 deterministic summarization) that map a window of turns $(h_{t-k:t}) \mapsto s_t$ where s_t is the summarized
225 representation, while preserving routing decisions, which leaves Proposition 1 applicable.
226

227 3.3 ALGORITHM: ONEFLOW



249 Figure 2: An example to show how OneFlow framework works. The framework employs dual meta-
250 LLMs (One creative workflow designer and one workflow critic) with Monte Carlo Tree Search to
251 automatically design multi-agent workflows that suitable for single-agent execution for complex
252 tasks. The left panel shows how the first five rounds of workflow design and selection process work.
253 The right panel shows how workflow 6 is generated.
254

255 For a given task T (e.g., select a most relevant query to the customer from four candidate queries),
256 we seek to return an optimal LLM agentic workflow W^* that is suitable for this task T . Ideally, the
257 optimal workflow W^* should be able to solve the task T with high performance and low cost. We
258 formulate this as a multi-objective optimization problem:
259

$$W^* = \arg \max_{W \in \mathcal{W}} [\alpha \cdot P(W, T) - \beta \cdot C(W, T)] \quad (1)$$

260 where $P(W, T)$ represents the agentic workflow W 's performance on task T (e.g., accuracy),
261 $C(W, T)$ denotes the operational cost (inference cost or token consumption), and α, β are balancing
262 hyperparameters that reflect the relative importance of performance and cost.
263

264 We approach this as a search problem within a search space that includes LLM agents (LLMs with
265 specific system prompts and tools) and their communication patterns (such as sequential execution,
266 conditional execution, and loops). This creates a discrete and potentially infinite search space. The
267 most common way to address this is using human expert priors to design specific workflows, e.g.,
268 Camel (Li et al., 2023), OAgent (Zhu et al., 2025), ReAct (Yao et al., 2023), LLM debate (Du
269 et al., 2023), etc. For automatic methods, there have been methods to automatically optimize the
prompt (Khattab et al., 2024) and also automatically design the workflow (Zhang et al., 2025e;b).

270 Following AFlow (Zhang et al., 2025e), we adopt Monte Carlo Tree Search (MCTS) to search this
 271 discrete search space. The entire search process follows these steps (illustrated in Figure 2):
 272

273 Algorithm 1 OneFlow (MCTS with Dual Meta-LLMs)

274 **Require:** Task T , validation set D_v , iterations K , weights α, β
 275 **Ensure:** Best workflow W^*

276 1: $W_{\text{IO}} \leftarrow$ single-LLM input-output; $\text{Eval}(W_{\text{IO}} \text{ on } D_v)$; init tree with W_{IO}
 277 2: **for** $k = 1$ to K **do**
 278 3: $\mathcal{C} \leftarrow \{W_{\text{IO}}\} \cup \text{Top-}(n-1) \text{ by score } S(W) = \alpha P - \beta C$ ▷ Selection; App. A.2
 279 4: $W \leftarrow \text{Select}(\mathcal{C})$
 280 5: $W_d \leftarrow \text{Designer}(W, \text{errors}(W))$ ▷ Meta-LLM 1; App. A.3
 281 6: $W_{\text{new}} \leftarrow \text{Reviewer}(W_d, \mathcal{C})$ ▷ Meta-LLM 2
 282 7: $\text{Eval}(W_{\text{new}} \text{ on } D_v)$; attach W_{new} ; backpropagate
 283 8: **end for**
 284 9: **return** $\arg \max_W \alpha P(W, T) - \beta C(W, T)$ over explored nodes

285 *Initialization.* We begin with the simplest possible workflow: $W_{\text{Input-Output}}$, which directly uses a
 286 single agent to answer questions in a straightforward input-output manner. We evaluate this work-
 287 flow on a validation dataset D_v (where $|D_v|$ means 20% samples of a specific dataset, e.g., the
 288 HumanEval dataset has 164 questions, so the validation dataset has 33 questions) to measure its
 289 performance, including both performance metrics (such as accuracy) and cost metrics (such as in-
 290 ference cost in USD). This input-output workflow becomes the root node in our Monte Carlo tree
 291 (workflow 1 in Figure 2), storing both performance metrics and examples of failure cases.

292 *Iterative Monte Carlo Tree Search (MCTS) Process.* We then follow a standard four-stage MCTS ap-
 293 proach, adapted for workflow optimization. 1. *Selection.* Choose a workflow W from the tree based
 294 on a performance-based probability distribution. 2. *Expansion.* Based on the selected workflow W ,
 295 generate a new workflow W_{new} using dual meta-LLMs that collaboratively design improved work-
 296 flows. 3. *Evaluation.* Test the new workflow W_{new} on validation data to obtain performance and
 297 cost metrics. 4. *Backpropagation.* Compare the performance and cost metrics of the new workflow
 298 W_{new} with the parent workflow W , and record this modification to the parent workflow W to avoid
 299 redundant designs. This iterative process continues until reaching a maximum number of iterations
 300 (we set it to 20 iterations in our experiments), producing a diverse set of automatically generated
 301 workflows tailored to the specific task.

302 *Dual Meta-LLM Architecture.* As illustrated in Figure 2, OneFlow employs two specialized meta-
 303 LLMs during the expansion phase: a Creative Designer that proposes performance-focused work-
 304 flow improvements, and a Critical Reviewer that refines these proposals to optimize cost-efficiency.
 305 This collaborative design balances the performance-cost trade-off in Equation 1. Details about al-
 306 gorithm can be found in the appendix MCTS optimization (Section A.2) and dual meta-LLMs for
 307 balanced performance and cost optimization (Section A.3).

308 **OneFlow includes two stages: first, it searches for the optimized workflow, then performs**
 309 **single LLM implementation of the optimized workflow.** We use OneFlow and OneFlow (single-
 310 agent execution) to clearly distinguish these two phases.

311 **4 EXPERIMENTS**

312 **4.1 EXPERIMENTAL SETUP**

313 **Datasets.** We evaluate our approach across a diverse set of benchmarks spanning multiple domains
 314 to assess generalization capabilities. Our evaluation suite includes: (i) *code generation tasks*: MBPP
 315 and HUMAN EVAL; (ii) *mathematical reasoning*: GSM8K and MATH; (iii) *question answering*:
 316 HOTPOTQA and DROP; (iv) *domain-specific reasoning*: SHOPPING-MMLU; and (v) *real-world*
 317 *planning and tool use*: TRAVELPLANNER.

318 **Evaluation Metrics.** We assess both task performance and computational efficiency. For code gen-
 319 eration, we report `pass@1` accuracy; for general question answering, we use F1 score; for mathe-
 320 matical tasks, we report solve rate (%); for Shopping-MMLU, we use accuracy. **For TravelPlanner,**
 321 **we use task success rate (%).** Computational cost is measured as USD token expenditure per work-

324 flow, accounting for both input and output token usage. To ensure statistical reliability, we conduct
 325 three independent trials and report mean values with standard deviations.
 326

327 **Baseline Methods.** We compare against four categories of approaches: (1) *Manual baselines*:
 328 Direct input-output (IO) prompting, chain-of-thought (CoT) prompting (Wei et al., 2022), **self-**
 329 **consistency (CoT)** and **MultiPersona** (Wang et al., 2024b); (2) *Automated multi-agent frameworks*:
 330 AFlow (Zhang et al., 2025e) and our proposed OneFlow; (3) *Heterogeneous multi-agent systems*:
 331 AFlow-optimized workflows using GPT-4o-mini and Claude-3.5-Haiku as heterogeneous executors;
 332 (4) *Single-LLM implementations*: Following Section 3.2, we execute multi-agent workflows (AFlow
 333 and OneFlow) using a single LLM agent.
 334

335 **Model Configuration.** Following established practices (Zhang et al., 2025e), we use GPT-4o-mini
 336 as the primary executor LLM across all methods with temperature set to 0. For robustness validation,
 337 we additionally evaluate with Claude-3.5-Haiku (results are in the appendix) **and Qwen-3 8B (to**
 338 **verify findings on open-weight models)**. Methods requiring workflow optimization (e.g., AFlow)
 339 employ Claude-4.0-Sonnet as the designer/optimizer with 20 optimization rounds.
 340

341 **Heterogeneous Multi-agent Workflow Implementation.** To investigate model heterogeneity ef-
 342 fects, we leverage AFlow (Zhang et al., 2025e) to automatically design heterogeneous multi-agent
 343 workflows. For homogeneous multi-agent workflows, all the executor LLMs are the same (either
 344 gpt-4o-mini or claude 3.5 haiku). In this heterogeneous setting, the workflow has two executor mod-
 345 els at the same time, GPT-4o-mini and Claude-3.5-Haiku. Claude-4.0-Sonnet serves as the workflow
 346 designer. Optimization iterations are capped at 20 rounds with temperature 0. API interactions uti-
 347 lize OpenAI Chat Completions and Anthropic interfaces. System prompts for the optimizing process
 348 are detailed in Appendix A.6.
 349

350 **KV Cache and Cost Estimation for Single-agent Implemented Multi-agent Workflows.** Im-
 351 plementing KV-cache optimization (Section 3.2) typically requires open-weight LLMs. Since we
 352 employ closed-weight LLMs (GPT-4o-mini) via API calls, we simulate ideal KV-cache costs by uti-
 353 lizing the final conversation states (the final message list). Cost calculations follow OpenAI’s official
 354 tokenization for GPT-4o-mini, categorizing user prompts as input tokens and assistant responses as
 355 output tokens to estimate the theoretical KV-cache cost. **For open-weight models (Qwen-3 8B), we**
 356 **explicitly measure latency and throughput using vLLM with KV cache enabled.**
 357

358 4.2 EXPERIMENTAL RESULTS AND ANALYSIS

359 Table 1: Main results on public benchmarks with GPT-4o mini as executors. Values are mean \pm std
 360 over three runs, in percentage (0–100). Code tasks report pass@1; QA tasks report F1; solve rate
 361 (%) for math. Best per column in **bold**; runner-up per column underlined.
 362

363 Method	CODE		MATH		QA	
	364 HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP
Manual baselines						
365 IO	89.1 \pm 0.4	72.6 \pm 0.3	87.0 \pm 0.1	51.3 \pm 0.6	71.1 \pm 0.5	66.2 \pm 0.6
366 CoT (Wei et al., 2022)	90.3 \pm 1.2	73.3 \pm 0.0	87.1 \pm 0.2	50.9 \pm 0.3	71.2 \pm 1.0	78.9 \pm 0.4
367 CoT SC (5-shot)	89.8 \pm 0.9	71.9 \pm 1.0	92.6 \pm 0.8	37.7 \pm 1.2	67.3 \pm 0.3	79.4 \pm 0.1
368 MultiPersona (Wang et al., 2024b)	89.1 \pm 0.2	73.3 \pm 0.3	87.1 \pm 0.1	50.9 \pm 0.3	71.2 \pm 1.0	78.9 \pm 0.4
Automatically designed multi-agent frameworks						
369 AFlow (Zhang et al., 2025e)	90.1 \pm 0.0	78.8 \pm 0.7	93.6 \pm 0.5	55.6 \pm 0.3	72.1 \pm 0.2	83.1 \pm 0.3
370 OneFlow	91.6 \pm 0.8	81.1 \pm 0.4	93.0 \pm 0.4	53.4 \pm 1.4	73.5 \pm 0.5	81.1 \pm 0.8
Single-LLM implementation of multi-agent workflow						
371 AFlow (single-agent execution)	91.1 \pm 1.6	78.8 \pm 0.7	92.9 \pm 0.1	53.8 \pm 0.9	68.4 \pm 0.1	81.1 \pm 0.7
372 OneFlow (single-agent execution)	92.1 \pm 0.4	81.4 \pm 0.6	<u>93.3 \pm 0.1</u>	<u>54.1 \pm 0.7</u>	<u>73.5 \pm 0.5</u>	<u>81.7 \pm 0.7</u>

373 4.2.1 PERFORMANCE ON PUBLIC BENCHMARKS

374 **A single-agent implementation of a multi-agent workflow can match multi-agent performance.**
 375 Table 1 summarizes results with GPT-4o mini (pass@1 for code; F1 for QA; solve rate (%)) for math).
 376 Across the board, automatically designed multi-agent workflows and their single-agent executions
 377 substantially outperform manual baselines, highlighting the value of automated design. Notably, ex-
 378 ecuting AFlow- and OneFlow-designed workflows with *a single agent* matches or slightly exceeds

378 their multi-agent counterparts, consistent with the hypothesis in Section 3.1: in homogeneous settings,
 379 a single model can faithfully simulate agent roles via multi-turn conversations. Our proposed
 380 OneFlow, typically when using the single-agent execution setting, shows superior performance com-
 381 pared to all the homogeneous workflow baselines.

383 4.2.2 COST ON PUBLIC BENCHMARKS

385 Table 2: Inference cost (USD) on public benchmarks with GPT-4o mini. Values are mean \pm std over
 386 three runs (lower is better). Best per column in **bold**; runner-up per column underlined.

Method	CODE		MATH		QA	
	HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP
<i>Manual baselines</i>						
CoT SC (5-shot)	\$0.103 \pm 0.000	\$0.177 \pm 0.000	\$1.265 \pm 0.001	\$1.561 \pm 0.009	\$1.201 \pm 0.001	\$0.613 \pm 0.001
MultiPersona	\$0.099 \pm 0.000	\$0.226 \pm 0.001	<u>\$0.429 \pm 0.005</u>	\$0.330 \pm 0.021	\$0.415 \pm 0.000	<u>\$0.301 \pm 0.000</u>
<i>Automatically designed multi-agent frameworks</i>						
AFlow	\$0.198 \pm 0.003	\$0.393 \pm 0.002	\$1.134 \pm 0.001	\$2.343 \pm 0.036	\$1.438 \pm 0.000	\$0.771 \pm 0.001
OneFlow	<u>\$0.026 \pm 0.000</u>	<u>\$0.070 \pm 0.005</u>	\$0.623 \pm 0.001	\$0.819 \pm 0.007	\$0.278 \pm 0.000	\$0.322 \pm 0.000
<i>Single-LLM implementation of multi-agent workflow</i>						
AFlow (single-agent execution)	\$0.198 \pm 0.004	\$0.283 \pm 0.001	\$0.697 \pm 0.001	\$2.039 \pm 0.028	\$0.530 \pm 0.001	\$0.345 \pm 0.001
OneFlow (single-agent execution)	\$0.020 \pm 0.000	<u>\$0.063 \pm 0.004</u>	\$0.387 \pm 0.000	<u>\$0.677 \pm 0.002</u>	\$0.278 \pm 0.000	\$0.284 \pm 0.001

391 **Single-agent execution is substantially more efficient and cheaper than multi-agent execution.**
 392 Table 2 shows that single-LLM execution dramatically reduces cost at comparable performance
 393 (Table 1), largely due to KV-cache reuse across agent turns in homogeneous workflows. Without
 394 single-LLM execution, OneFlow is more cost-efficient than AFlow; when executed as a single LLM,
 395 both AFlow and OneFlow realize cost gains and still maintain performance. For OneFlow, the
 396 performance even slightly increases with single-agent execution, thanks to KV sharing, which provides
 397 more context to the agent and generates better results. Table 10 further breaks down input/output
 398 tokens and explains where the savings arise.

404 Table 3: Executor-specific results and heterogeneous baseline on public benchmarks. Values are
 405 mean \pm std over three runs, in percentage (0–100). Best per column in **bold**; best per base model
 406 type per column underlined.

Method (Executor)	CODE		MATH		QA	
	HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP
<i>GPT-4o mini-based</i>						
AFlow (GPT-4o mini)	90.1 \pm 0.0	78.8 \pm 0.7	93.6 \pm 0.5	55.6 \pm 0.3	72.1 \pm 0.2	<u>83.1 \pm 0.3</u>
OneFlow (GPT-4o mini)	<u>91.6 \pm 0.8</u>	81.1 \pm 0.4	93.0 \pm 0.4	53.4 \pm 1.4	<u>73.5 \pm 0.5</u>	81.1 \pm 0.8
AFlow (GPT-4o mini, single-agent)	91.1 \pm 1.6	78.8 \pm 0.7	92.9 \pm 0.1	53.8 \pm 0.9	68.4 \pm 0.1	81.1 \pm 0.7
OneFlow (GPT-4o mini, single-agent)	92.1 \pm 0.4	<u>81.4 \pm 0.6</u>	93.3 \pm 0.1	54.1 \pm 0.7	<u>73.5 \pm 0.5</u>	81.7 \pm 0.7
<i>Claude 3.5 Haiku-based</i>						
AFlow (Claude 3.5 Haiku)	90.8 \pm 0.0	83.6 \pm 0.0	91.2 \pm 0.0	50.5 \pm 0.0	74.6 \pm 0.0	86.8 \pm 0.0
OneFlow (Claude 3.5 Haiku)	<u>91.6 \pm 0.0</u>	84.4 \pm 0.0	<u>93.0 \pm 0.0</u>	<u>51.3 \pm 0.0</u>	<u>74.7 \pm 0.0</u>	87.5 \pm 0.0
<i>Heterogeneous baseline</i>						
AFlow (Heterogeneous: GPT-4o mini + Claude 3.5 Haiku)	87.0 \pm 0.8	80.0 \pm 0.3	93.6 \pm 0.3	55.7 \pm 0.6	75.1 \pm 0.5	85.5 \pm 0.5

419 4.2.3 PILOT STUDY ON AUTOMATICALLY DESIGNED HETEROGENEOUS MULTI-AGENT 420 WORKFLOWS

421 **Performance is largely bounded by the best homogeneous workflow.** We conduct a pilot study
 422 using an automatically designed heterogeneous multi-agent workflow with GPT-4o mini and Claude
 423 3.5 Haiku collaboratively working within the workflow. We notice that the performance of this
 424 heterogeneous workflow is largely bounded by the homogeneous multi-agent workflow. E.g., the
 425 performance on DROP achieves an F1-score of 85.5, even outperforming all GPT-4o mini-based
 426 methods (with AFlow utilizing GPT-4o mini having an F1 of 83.1), but is still bounded by the best
 427 performance of the Claude 3.5 Haiku-based homogeneous workflow (specifically OneFlow, with
 428 an F1 of 87.5). However, we want to emphasize that this pilot study uses automatically generated
 429 heterogeneous multi-agent workflows, which are not perfectly optimized. When deploying multi-
 430 agent systems in the real world, **well-designed** heterogeneous multi-agent workflows can be very
 431 beneficial. For example, for simple tasks, we can use a small LLM agent to handle them, while
 432 for complex tasks, we may route them to a strong reasoning model. **Implications.** In homogeneous

settings, single-agent execution is a strong, cost-efficient baseline and even matched the performance of AFlow-optimized heterogeneous workflows. Two directions stand out: (1) train a single agent to execute complex workflows end-to-end; (2) design effective heterogeneous workflows that mix models of different strengths and costs to collaborate efficiently despite no KV sharing.

4.2.4 KV CACHE REUSE WITH OPEN-WEIGHT LLMs

Table 4: Results on HumanEval with Qwen-3 8B. Single-agent execution maintains performance and efficiency thanks to KV cache reuse. Scores are averaged over three independent runs.

Method	pass@1	Avg Latency (s)	Throughput (samples/s)	Avg Input Tokens	Avg Output Tokens
CoT	83.5%	2.60	7.66	213	74
CoT SC $\times 5$	83.7%	17.44	1.32	1748	502
MultiPersona	84.7%	32.38	0.75	1722	846
AFlow (multiple stateless api calls)	86.8%	54.98	0.17	2302	1753
AFlow (Single-agent execution)	90.5%	53.53	0.18	3269	1739
OneFlow (multiple stateless api calls)	87.0%	4.31	2.47	920	148
OneFlow (Single-agent execution)	87.4%	4.83	1.70	1288	159

To validate our findings beyond proprietary models, we conduct experiments using the open-weight Qwen-3 8B model with vLLM, setting the context window to 16k to reflect typical resource constraints. As shown in Table 4, single-agent execution of both AFlow and OneFlow maintains or improves performance (pass@1 on HumanEval) compared to multiple stateless API calls. Crucially, while multi-turn conversations increase the average input tokens due to history accumulation (+967 for AFlow, +368 for OneFlow), the inference efficiency (latency and throughput) remains stable thanks to KV cache reuse. We note that Qwen-3 8B tends to generate longer responses in multi-turn settings compared to single-turn, slightly offsetting efficiency gains seen with GPT-4o-mini, but the core benefit of KV cache sharing in homogeneous workflows is confirmed.

4.2.5 ADDITIONAL EXPERIMENTS ON THE TOOL-USING BENCHMARK: TRAVELPLANNER

We additionally evaluate on TravelPlanner (Xie et al., 2024), a context- and tool-intensive benchmark for real-world planning, to assess whether single-agent execution of homogeneous multi-agent workflows remains competitive. A single LLM executing the AFlow and OneFlow workflows matches the task success rate of their original multi-agent counterparts while incurring lower inference cost. The workflow uncovered by OneFlow is especially compact and efficient, further reducing cost. The results are shown in Figure 3.

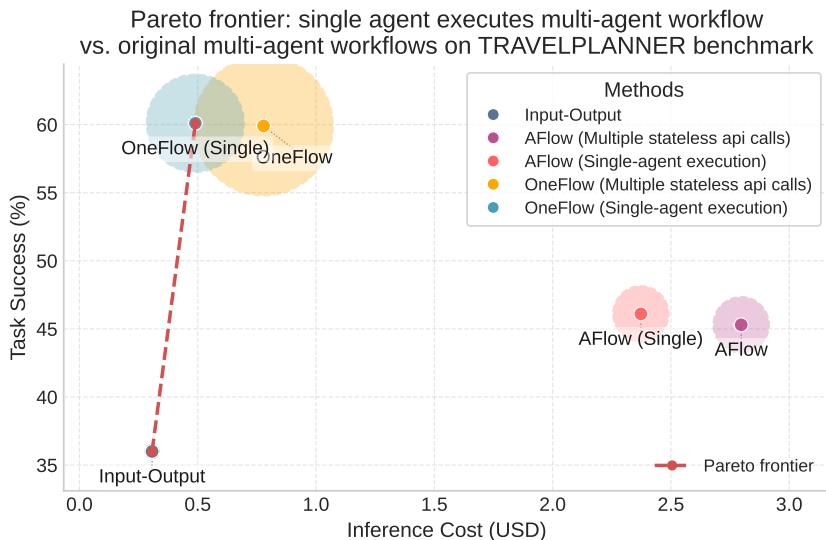


Figure 3: Pareto frontier: single agent executes multi-agent homogeneous workflow vs. original multi-agent workflows on TravelPlanner (Xie et al., 2024) benchmark. All the workflows are searched by Claude 4 Sonnet. All the workflows are executed by GPT-4o mini.

486 5 RELATED WORK

488 **Multi-Agent Workflows and Task Decomposition.** Multi-agent systems organize multiple LLM
 489 agents with complementary roles and communication patterns to solve complex tasks more reli-
 490 ably than a single-pass input-output baseline. Foundational designs include role-playing and tool-
 491 augmented collaboration (e.g., CAMEL, OAgent, ReAct, debate) that decompose tasks into spe-
 492 cialized stages and coordinate agents via sequential, iterative, or deliberative interactions (Li et al.,
 493 2023; Zhu et al., 2025; Yao et al., 2023; Du et al., 2023). A key empirical trend is that *most* frame-
 494 works are **homogeneous**: agents share the same base LLM and differ by prompts, tools, and routing
 495 logic (Zhuge et al., 2024; Liu et al., 2024). Heterogeneous systems, where agents use different base
 496 models, have been explored to leverage model diversity (Ye et al., 2025a; Zhang et al., 2025a; Wang
 497 et al., 2024a; Jiang et al., 2023; Chen et al., 2023), but many such approaches emphasize ensem-
 498 bling or discussion rather than end-to-end **execution** with explicit control flow. This paper examines
 499 when a single agent can faithfully simulate homogeneous workflows via multi-turn conversations,
 500 and clarifies where heterogeneity may bring benefits and costs.

501 **Automatic Design of Multi-Agent Workflows.** Reducing human effort in workflow construction
 502 has led to three complementary directions: (1) *prompt optimization*, (2) *communication/topology*
 503 *optimization*, and (3) *automatically workflow search/generation*. For prompt optimization, DSPy
 504 formalizes modular prompt programs with compilation-time optimization, and TextGrad proposes
 505 gradient-inspired improvements (Khattab et al., 2024; Yuksekgonul et al., 2025). For communi-
 506 cation/topology optimization, GPTSwarm explores graph-structured agent teams with iterative re-
 507 finement; Dylan performs dynamic agent selection; Agent-Prune prunes redundant edges; and G-
 508 Designer learns task-adaptive topologies (Zhuge et al., 2024; Liu et al., 2024; Zhang et al., 2025c;d).
 509 For automatic workflow design, ADAS is the first to propose this idea and conducts heuristic
 510 search; AFlow uses MCTS with named nodes; MaAS optimizes a distribution over architectures
 511 via a controller; AgentSquare extends this paradigm; and recent methods generate workflows di-
 512 rectly via fine-tuning or continuous optimization (MAS-GPT, Flow-reasoner, ScoreFlow), or train
 513 weaker meta-agents to design for stronger executors (Weak-for-strong) (Hu et al., 2025; Zhang et al.,
 514 2025e;b; Shang et al., 2025; Ye et al., 2025b; Gao et al., 2025; Wang et al., 2025b; Nie et al., 2025).
 515 While these systems substantially reduce manual design effort, the vast majority assume homo-
 516 geneous executors; some incorporate heterogeneous options only tangentially (e.g., MAS-Router,
 517 limited heterogeneity settings in AFlow) (Yue et al., 2025; Zhang et al., 2025e). Our work comple-
 518 ments this line by showing that a strong single-LLM simulator provides a competitive, cost-efficient
 519 baseline for many automatically designed *homogeneous* workflows, while clarifying when hetero-
 520 geneity remains necessary. We also note an orthogonal trend toward smaller, cost-efficient models
 521 that further motivates cost-aware workflow design (Belcak et al., 2025).

522 **KV Cache and Single-LLM Execution.** Transformer KV caching reuses previously computed
 523 key/value states to avoid repeated prefill, enabling substantial speedups in autoregressive decoding.
 524 When multiple agents share the *same* base model (homogeneous workflows), a single-LLM execu-
 525 tion can maintain one conversation with shared KV cache, avoiding redundant re-encoding across
 526 agent turns and often improving consistency. In contrast, *heterogeneous* workflows cannot share KV
 527 states across different base models, limiting these gains and complicating end-to-end training. Prior
 528 work has analyzed caching behaviors and memory allocation (e.g., attention sinks) and proposed
 529 structured caches for long or graph-structured contexts (Xiao et al., 2024; Wang et al., 2025a). These
 530 observations motivate our study: if behavior can be preserved (under mild conditions) and caches
 531 can be shared, single-LLM simulations offer an informative and strong baseline for homogeneous
 532 agentic workflows.

533 6 CONCLUSION

534 In homogeneous multi-agent workflows, a single LLM can role-play agents via multi-turn conver-
 535 sations, reuse a shared KV cache, and match or slightly exceed multi-agent performance at substan-
 536 tially lower cost. **OneFlow** helps discover compact workflows and, when executed by a single LLM,
 537 yields additional efficiency gains. While single-LLM simulation cannot realize true heterogeneity,
 538 our pilot shows it can even match the performance of AFlow-optimized heterogeneous workflows;
 539 we view (1) **training single agents for end-to-end execution** and (2) **principled heterogeneous**
 540 **composition** as complementary, promising directions in LLM multi-agent system research.

540 ETHICS STATEMENT

541
 542 This research primarily focuses on evaluating the effectiveness of using a single LLM (large lan-
 543 guage model) agent to perform multi-agent workflows, with an emphasis on empirical validation. It
 544 relies solely on existing LLMs and does not involve training, fine-tuning model weights, or creating
 545 new LLMs. As such, the work does not raise any novel ethical considerations or societal impacts
 546 beyond those already well documented in relation to large-scale language models more broadly.

547 REPRODUCIBILITY STATEMENT

548
 549 We provide pseudocode and a detailed explanation of OneFlow in the main methodology section,
 550 along with a visual illustration for clarity. We also include the exact prompts used, as well as the
 551 model settings for both the executor model and the optimization model.

553 REFERENCES

554
 555 Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan,
 556 Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic
 557 ai, 2025. URL <https://arxiv.org/abs/2506.02153>.

558
 559 Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference
 560 improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*, 2023.

561
 562 Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving
 563 factuality and reasoning in language models through multiagent debate, 2023. URL <https://arxiv.org/abs/2305.14325>.

564
 565 Hongcheng Gao, Yue Liu, Yufei He, Longxu Dou, Chao Du, Zhijie Deng, Bryan Hooi, Min Lin,
 566 and Tianyu Pang. Flowreasoner: Reinforcing query-level meta-agents, 2025. URL <https://arxiv.org/abs/2504.15257>.

567
 568 Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth*
 569 *International Conference on Learning Representations*, 2025.

570
 571 Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models
 572 with pairwise ranking and generative fusion, 2023. URL <https://arxiv.org/abs/2306.02561>.

573
 574 Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful
 575 Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, et al. Dspy: Compil-
 576 ing declarative language model calls into state-of-the-art pipelines. In *The Twelfth International*
 577 *Conference on Learning Representations*, 2024.

578
 579 Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbulin, and Bernard Ghanem.
 580 Camel: communicative agents for "mind" exploration of large language model society. In *Pro-
 581 ceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS
 582 '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

583
 584 Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network
 585 for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024.

586
 587 Fan Nie, Lan Feng, Haotian Ye, Weixin Liang, Pan Lu, Huaxiu Yao, Alexandre Alahi, and James
 588 Zou. Weak-for-strong: Training weak meta-agent to harness strong executors. *arXiv preprint*
 589 *arXiv:2504.04785*, 2025.

590
 591 Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic
 592 LLM agent search in modular design space. In *The Thirteenth International Conference on Learn-
 593 ing Representations*, 2025. URL <https://openreview.net/forum?id=mPdmDYIQ7f>.

594
 595 Haoyu Wang, Peihao Wang, Mufei Li, Shikun Liu, Siqi Miao, Zhangyang Wang, and Pan Li. Graph-
 596 kv: Breaking sequence via injecting structural biases into large language models. *arXiv preprint*
 597 *arXiv:2506.07334*, 2025a.

594 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha
 595 Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language
 596 models. In *The Eleventh International Conference on Learning Representations*, 2023.

597

598 Yinjie Wang, Ling Yang, Guohao Li, Mengdi Wang, and Bryon Aragam. Scoreflow: Mastering
 599 llm agent workflows via score-based preference optimization. *arXiv preprint arXiv:2502.04306*,
 600 2025b.

601 Zhengdong Wang, Zhenghao Xu, Xun Jin, and Ge Dong. Mixture-of-agents enhances large language
 602 model capabilities. *arXiv preprint arXiv:2406.04692*, 2024a.

603

604 Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing
 605 the emergent cognitive synergy in large language models: A task-solving agent through multi-
 606 persona self-collaboration. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings*
 607 *of the 2024 Conference of the North American Chapter of the Association for Computational*
 608 *Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 257–279, Mexico
 609 City, Mexico, June 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.
 610 naacl-long.15. URL <https://aclanthology.org/2024.naacl-long.15/>.

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

614 Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming
 615 language models with attention sinks. In *The Twelfth International Conference on Learning Rep-*
 616 *resentations*, 2024.

617 Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and
 618 Yu Su. Travelplanner: a benchmark for real-world planning with language agents. In *Proceedings*
 619 *of the 41st International Conference on Machine Learning*, pp. 54590–54613, 2024.

620

621 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.
 622 React: Synergizing reasoning and acting in language models, 2023. URL <https://arxiv.org/abs/2210.03629>.

623

624 Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. X-
 625 mas: Towards building multi-agent systems with heterogeneous llms, 2025a. URL <https://arxiv.org/abs/2505.16997>.

626

627 Rui Ye, Shuo Tang, Rui Ge, Yixin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. MAS-GPT: Train-
 628 ing LLMs to build LLM-based multi-agent systems. In *Forty-second International Conference on*
 629 *Machine Learning*, 2025b. URL <https://openreview.net/forum?id=3CiSpY3QdZ>.

630

631 Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan
 632 Qi. Masrouter: Learning to route llms for multi-agent systems. *arXiv preprint arXiv:2502.11133*,
 633 2025.

634

635 Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin,
 636 and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*,
 637 639:609–616, 2025.

638

639 Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue
 640 Hu, and Lei Bai. Evoflow: Evolving diverse agentic workflows on the fly. *arXiv preprint*
 641 *arXiv:2502.07373*, 2025a.

642

643 Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent
 644 architecture search via agentic supernet. In *Forty-second International Conference on Machine*
 645 *Learning*, 2025b. URL <https://openreview.net/forum?id=imcyV1zpXh>.

646

647 Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng,
 648 Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for
 649 llm-based multi-agent systems. In *The Thirteenth International Conference on Learning Repre-*
 650 *sentations*, 2025c.

648 Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang,
 649 Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topolo-
 650 gies via graph neural networks. In *Forty-second International Conference on Machine Learning*,
 651 2025d.

652 Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen
 653 Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow genera-
 654 tion. In *The Thirteenth International Conference on Learning Representations*, 2025e.

655 He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li,
 656 Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang,
 657 Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, Jian Yang, Minghao Liu, Xitong Gao,
 658 Jiaheng Liu, and Wangchunshu Zhou. Oagents: An empirical study of building effective agents,
 659 2025. URL <https://arxiv.org/abs/2506.15741>.

660 Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbulin, and Jürgen
 661 Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *Forty-first International*
 662 *Conference on Machine Learning*, 2024.

663 A APPENDIX

664 A.1 THE USE OF LARGE LANGUAGE MODELS

665 Large language models were used solely for sentence-level proofreading. All research ideation and
 666 paper writing were conducted entirely by the authors.

667 A.2 MONTE CARLO TREE SEARCH FOR WORKFLOW OPTIMIZATION

668 As mentioned above and Figure 2, we employ MCTS to systematically explore the space of possible
 669 workflows, treating each workflow configuration as a node in the search tree. Specifically:

670 **1. Selection:** At each iteration, we form a candidate set consisting of the initial workflow
 671 $W_{\text{Input-Output}}$ together with the top 3 best-performing workflows from the Monte Carlo Tree. Retain-
 672 ing the initial workflow $W_{\text{Input-Output}}$ in the candidate set helps the framework maintain exploration
 673 of the workflow space. We then select one Workflow W workflow from the candidate set for ex-
 674 pansion. Following AFlow (Zhang et al., 2025e), we employ a mixed-probability selection strategy that
 675 combines uniform distribution (for exploration) and score-based weighting (for exploitation):

$$676 p_i = \lambda \cdot \frac{1}{n} + (1 - \lambda) \cdot \frac{\exp(\alpha \cdot s_i)}{\sum_j \exp(\alpha \cdot s_j)} \quad (2)$$

677 where s_i is the score of workflow i , α controls the sharpness of the distribution, λ balances explo-
 678 ration and exploitation, and $n \leq 4$ represents the number of candidates. This formula ensures that
 679 better-performing workflows have a higher probability of being selected as Workflow W .

680 **2. Expansion:** This phase leverages the dual meta-LLM architecture (detailed in Appendix A.3).
 681 At each expansion, the designer meta-LLM proposes a new workflow design W_{new1} (represented
 682 in Python code with detailed comments) based on the selected workflow W from the previous se-
 683 lection step and the failed cases in the validation dataset D_v . Next, the critic meta-LLM reviews the
 684 proposed workflow design W_{new1} to assess its validity and efficiency by examining the code com-
 685 ments and Python implementation. The critic also compares the proposal W_{new1} with candidate
 686 workflows from the selection stage. Finally, the critic meta-LLM proposes an improved workflow
 687 W_{new2} based on the designer’s proposal W_{new1} , writing detailed comments in the workflow code
 688 to record changes and their rationale. The output of this stage is a runnable workflow.py file that
 689 implemented W_{new2} .

690 **3. Evaluation.** The proposed workflow W_{new2} is then evaluated on the validation dataset D_v
 691 to obtain performance metrics (e.g., accuracy), cost metrics (e.g., token consumption), and failed
 692 samples (including the specific reasoning processes of the failures).

702 **4. Backpropagation:** The performance and cost, and failure cases are stored in the current workflow
 703 node W_{new2} . The relative success or failure compared to the parent workflow node W is stored
 704 in the parent node W (we call this backpropagation) to avoid proposing similar workflows. The
 705 optimization process continues until a maximum number of iterations is reached (20).
 706

707 **A.3 DUAL META-LLMs FOR BALANCED PERFORMANCE AND COST OPTIMIZATION**
 708

709 We design two specialized meta-LLMs that collaboratively design new workflows (the right panel
 710 of Figure 2 shows how this dual meta-LLM framework works):

711 **Meta-LLM 1: Creative Designer.** This meta-LLM’s goal is to creatively, even disruptively, im-
 712 prove a workflow. It receives the code and prompt of the current agentic workflow $W_{current}$, the
 713 workflow’s performance score (e.g., accuracy), any previous modifications based on $W_{current}$, the
 714 corresponding performance score, and a sample of incorrectly answered questions, D_{error} , including
 715 the questions, the reasoning process when the workflow failed to answer the question, and the ground
 716 truth answer. It also receives instructions on how to write a runnable `workflow.py` file, how to
 717 define system prompts for LLM agents, and how to utilize existing operators (e.g., chain-of-thought
 718 reasoning, ensembling, executing Python code, etc.). With this information, the Creative Designer
 719 is prompted to make creative modifications to improve workflow performance and add comments to
 720 the workflow code explaining the rationale for the changes.

$$W_{creative\ designer} = \text{Designer}(W_{current}, D_{error}, \text{instructions}) \quad (3)$$

721 The creative designer’s setting is similar to the AFlow (Zhang et al., 2025e) setting, with two mod-
 722 ifications: 1) the creative designer is prompted to write code comments explaining the rationale
 723 for modifications, which can inform the other meta-LLM (the critical reviewer), and 2) the cre-
 724 ative designer is given more detailed error logs, which include the specific reasoning process during
 725 the entire workflow’s execution. This helps the creative designer better understand the workflow’s
 726 failure cases. Empirically, we found that the creative designer alone can propose workflows with
 727 excellent performance, but the inference cost of the workflow is high.

728 **Meta-LLM 2: Critical Reviewer.** The Critical Reviewer’s goal is to review and critique the cre-
 729 ative designer’s proposed workflow and modify it to avoid mistakes and improve cost-efficiency. To
 730 accomplish this, we provide the critical reviewer with the same information as the creative designer.
 731 Additionally, we give the critical reviewer the creative designer’s proposed workflow (a runnable
 732 `workflow.py` file, including code and comments) and the cost and performance metrics of exist-
 733 ing workflow candidates (the top 3 best-performing workflows from the Monte Carlo tree, plus the
 734 initial workflow $W_{Input-Output}$). We then prompt the critical reviewer to carefully examine the cre-
 735 ative designer’s proposed workflow, using the existing workflow candidates’ cost and performance
 736 as reference points. The critical reviewer proposes a new workflow design based on the designer’s
 737 proposal, writing detailed comments in the workflow code to document changes and their rationale.
 738 The goal is to avoid mistakes and improve cost-efficiency. The output of this stage is a runnable
 739 `workflow.py` file. The critical reviewer also writes reflections on the improved workflow. Since
 740 the critical reviewer is given more global information and cost data, it can provide a broader view of
 741 the workflow’s performance and cost trade-offs to guide future rounds of workflow design.

$$W_{critical\ reviewer} = \text{Reviewer}(W_{creative\ designer}, W_{candidates}, \text{cost_metrics}) \quad (4)$$

742 We use the critical reviewer’s output workflow as the final workflow of this round.
 743

744 **A.4 RESULTS ON CLAUDE 3.5 HAIKU**
 745

746 To further evaluate the effectiveness of our proposed OneFlow framework, we conduct experiments
 747 on Claude 3.5 Haiku. The results are shown in Table 5 and Table 6.
 748

749 **Performance on Shopping-Specific Tasks.** Table 7 demonstrates that our OneFlow approach
 750 achieves state-of-the-art performance across 9 out of 10 shopping-specific tasks, outperforming both
 751 single-LLM baselines and existing multi-agent frameworks. Notably, we observe substantial im-
 752 provements over the strongest baseline (AFlow) in critical shopping domains: Product Selection
 753 (+4.0% absolute improvement, 0.678 vs. 0.638), Sentiment Analysis (+2.5%, 0.777 vs. 0.752), and
 754 Multilingual Query Understanding (+4.8%, 0.569 vs. 0.521). These gains are particularly significant
 755 given the complexity of shopping-specific reasoning tasks, where domain knowledge and nuanced
 understanding of user intent are crucial.

756
 757 Table 5: Results on Claude 3.5 Haiku. Performance comparison on public benchmarks. We re-
 758 port pass@1 accuracy for code generation tasks (HumanEval, MBPP) and F1 scores for question-
 759 answering tasks (HotpotQA, DROP), solve rate (%) for GSM8K, MATH. OneFlow achieves com-
 760 petitive or superior performance across most benchmarks. Results are averaged over three indepen-
 761 dent runs.

Method	HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP
IO	0.870	0.745	0.875	0.329	0.635	0.640
AFlow (Zhang et al., 2025e)	0.908	0.836	0.912	0.505	0.746	0.868
OneFlow (Ours)	0.916	0.844	0.930	0.513	0.747	0.875

766
 767 Table 6: Results on Claude 3.5 Haiku. Cost comparison on public benchmarks. Values represent
 768 total inference cost in USD (lower is better). OneFlow achieves substantial cost reductions while
 769 maintaining competitive performance. Results are averaged over three independent runs.

Method	HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP
AFlow (Zhang et al., 2025e)	0.20	2.39	6.68	3.55	6.55	3.46
OneFlow (Ours)	0.25	0.58	5.83	4.33	1.47	3.14

770
 771 Table 7: Performance comparison on Shopping-MMLU tasks using accuracy as the evaluation met-
 772 ric. OneFlow achieves state-of-the-art performance across 9 out of 10 shopping-specific tasks. Re-
 773 sults are averaged over three independent runs. Task abbreviations: Product Sel. (applicable product
 774 selection), Sentiment (aspect-based sentiment classification), Attribute (implicit attribute selection),
 775 Multi-lang (multilingual query product semantic understanding), Prod. Comp. (product comple-
 776 ments), Query (query product semantic classification), Brand (related brands selection), Keyword
 777 (related keyword intent selection), Rev. Help (review helpfulness selection), Rev. Sent. (reviews
 778 overall sentiment selection). Best results are shown in **bold**.

Method	Product Sel.	Sentiment	Attribute	Multi-lang	Prod. Comp.	Query	Brand	Keyword	Rev. Help	Rev. Sent.
IO	0.638	0.668	0.650	0.465	0.713	0.464	0.681	0.728	0.586	0.578
CoT (Wei et al., 2022)	0.647	0.712	0.700	0.441	0.700	0.455	0.761	0.713	0.586	0.566
SC(CoT X 5) (Wang et al., 2023)	0.656	0.731	0.713	0.473	0.703	0.491	0.761	0.725	0.598	0.566
AFlow (Zhang et al., 2025e)	0.638	0.752	0.769	0.521	0.736	0.530	0.761	0.713	0.550	0.698
OneFlow (Ours)	0.678	0.777	0.775	0.569	0.756	0.567	0.756	0.783	0.609	0.718

789 Table 8: Cost efficiency comparison on Shopping-MMLU tasks measured in USD inference cost
 790 (lower is better). Results are averaged over three independent runs. OneFlow demonstrates superior
 791 cost efficiency across all shopping-specific tasks while maintaining competitive performance. Best
 792 results are shown in **bold**.

Method	Product Sel.	Sentiment	Attribute	Multi-lang	Prod. Comp.	Query	Brand	Keyword	Rev. Help	Rev. Sent.
SC(CoT X 5) (Wang et al., 2023)	1.87	1.78	1.62	1.84	1.89	1.55	1.34	2.00	1.53	2.78
AFlow (Zhang et al., 2025e)	1.47	1.83	2.20	1.32	1.39	0.63	0.23	0.39	0.86	2.75
OneFlow (Ours)	0.26	0.48	0.63	0.46	0.42	0.47	0.32	0.60	0.31	0.74

798 A.5 SYSTEM PROMPT FOR ONEFLOW

800 A.5.1 SYSTEM PROMPT FOR THE CREATIVE DESIGNER META-LLM

802 The following designer prompts are adapted from AFlow (Zhang et al., 2025e).

```
803
804 WORKFLOW_OPTIMIZE_PROMPT_DESIGNER = """You are building a workflow
805     Graph (nodes are llm agents, edges are the flow of information)
806     and corresponding Prompt to jointly solve {type} problems.
807     Referring to the given graph and prompt, which forms an example of a
808     {type} solution approach, please reconstruct and optimize them.
809     You can add, modify, or delete nodes, edges, or prompts. Include your
810     single modification in XML tags in your reply. Ensure they are
811     complete and correct to avoid runtime failures.
```

```

810 When optimizing, you can incorporate critical thinking methods such
811 as review, revise, ensemble (generating multiple answers through
812 different/similar prompts, then voting/integrating/checking the
813 majority to obtain a final answer), brainstorming, expert-based
814 reasoning, to solve the problem efficiently and effectively.
815 Consider Python's loops (for, while, list comprehensions),
816 conditional statements (if-elif-else, ternary operators), and
817 other programming techniques to enhance the workflow. Use logical
818 and control flow (IF-ELSE, loops) for a more enhanced graphical
819 representation.
820 You can design a workflow that adapts its approach based on the
821 complexity of each problem. For example, using Python if-else
822 statements to choose different solution strategies for easy vs.
823 hard problems.

824 PROMPT REQUIREMENTS:
825 - Only generate prompts used by Custom operators in your graph
826     (accessed as prompt_custom.YOUR_PROMPT_NAME)
827 - Built-in operators already have their own prompts - don't generate
828     prompts for them
829 - Remove any unused prompts from prompt_custom
830 - Generated prompts must be complete with no placeholders

831 Output the modified graph and all necessary prompt_custom prompts.
832 It's crucial to include necessary context during the process.
833 Be creative and try to push the boundaries of the existing graph.
834 Write concise and informative inline comments for the code you
835     modified to explain the logic and the purpose, why the
836     modification is creative and effective. With all the comments you
837     write, you should leave your name: Designer (modifying the round
838     X graph).
839 If you encounter code comments written by other designers or critics,
840     keep those that are useful for future reference.
841 """
842
843 WORKFLOW_INPUT_DESIGNER = """
844 Here is a workflow graph and the corresponding prompt (prompt only
845     related to the custom method) that has been tested on the
846     previous round and its score (maximum score is 1). You must make
847     further optimizations and improvements based on this graph. The
848     modified graph must differ from the provided example, and the
849     specific differences should be noted within the
850     <modification>xxx</modification> section.\n
851 <sample>
852     <experience>{experience}</experience>
853     <modification>(such as:add /delete /modify/ ...)</modification>
854     <score>{score}</score>
855     <graph>{graph}</graph>
856     <prompt>{prompt}</prompt>(contains prompts used by Custom
857     operators)
858
859     <operator_description>{operator_description}</operator_description>
860 </sample>
861
862 You are encouraged to use the operators provided, not limited to the
863     custom operator.

864 In the graph, you should use the list process to record the
865     intermediate output of the workflow. And return the process as
866     the one of the output of the workflow.
867 Below are the logs of some results with the aforementioned workflow
868     Graph encountered errors, which can be used as references for
869     optimization:

```

```

864 {log}
865
866 In your design, you cannot hardcode the answer (e.g., remember the
867 answer of a specific problem), which can lead to overfitting.
868 Instead, provide principles rather than specific answers.
869 First, provide optimization ideas. **Only one detail point can be
870 modified**, and no more than 5 lines of code may be changed per
871 modification; you can not totally change the existing graph,
872 extensive modifications are strictly prohibited to maintain
873 project focus!
874 When introducing new functionalities in the graph, please make sure
875 to import the necessary libraries or modules yourself, except for
876 operator, prompt_custom, create_llm_instance, which have already
877 been automatically imported.
878 No need to import the operator, prompt_custom, create_llm_instance,
879 which have already been automatically imported.
880
881 **Under no circumstances should Graph output None for any field.**
882 Use self.custom methods to restrict your output format, rather than
883 using code (outside of the code, the system will extract answers
884 based on certain rules and score them).
885 It is very important to format the Graph output answers, you can
886 refer to the standard answer format in the log.
887 Be creative and try your best to propose new ideas.
888
889 """
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

```

A.5.2 SYSTEM PROMPT FOR THE CRITICAL REVIEWER META-LLM

The following critic prompts are adapted from AFlow (Zhang et al., 2025e).

```

WORKFLOW_OPTIMIZE_PROMPT_CRITIC = """You are a great workflow critic
and optimizer.
You are the analytical critic and practical optimizer. Focus on (1)
the potential of the designer's idea to push the boundaries of
the existing workflow solutions and (2) cost-effectiveness of the
designer's proposed solution.
- The designer's idea has a great chance to help us push the
boundaries of the existing graph and may find great solutions, so
you should try to implement the designer's idea in a
cost-efficient way, if necessary.
- Preserve the designer's innovative ideas while making them more
cost-efficient, if necessary.
- You are encouraged to use the operators provided, not limited to
the custom operator.
- Fully respect the designer's idea, especially when the designer
proposed to use a predefined operators, you just use it and do
not modify it. Some operators can be expensive, but if the
designer proposed to use them, you should give them a try as it
may have great potential of performance improvement.
- If a solution involves too many api calls (agent) e.g., > 5 api
calls per problem and are not necessary, it can be very slow and
costly, you can try to find a way to make it more cost-efficient.
Overly expensive solutions are not allowed.
- You have access to the existing top performing workflow graphs and
their performance and cost metrics. Consider the performance and
cost of these existing graphs (they are the existing solutions
for the same problem) when optimizing the designer's graph. You
should not blindly reuse the existing top performing graphs, but
use them as references to improve the cost-efficiency of the
designer's idea, which may have great potential.

```

Code Commentary Requirements:

```

918 - Add concise inline comments explaining your critical analysis and
919   optimization rationale
920 - Include practical critiques at the code's end, drawing from
921   top-performing graphs and best practices
922 - Preserve useful existing comments from other designers/critics,
923   remove redundant ones
924 - Sign your comments as: "Critic (modifying round X workflow proposed
925   by the designer)"
926
927 ## Context and References:
928 When optimizing, refer to the designer's comments to understand the
929   logic, purpose, and creative value of each modification. Consider
930   how your optimization maintains the innovation while improving
931   practical efficiency.
932
933 Referring to the designer's given graph and prompt for {type}
934   problems, optimize them following the framework above."""
935
936
937 # Critic input prompts
938 WORKFLOW_INPUT_CRITIC = """
939 Here is a workflow graph and the corresponding prompt (prompt only
940   related to the custom method) that has been proposed by a
941   creative designer LLM (based on one existing workflow), which can
942   be a great idea. You must respect and implement the designer's
943   ideas. If the designer LLM's proposed workflow is already
944   cost-effective, you just use it and do not change it. When the
945   designer's idea is costly, you need to make further optimizations
946   based on this graph to make it more cost-efficient. Try to
947   preserve the original ideas as much as possible, but implement
948   them in a great cost-efficient way if necessary. You are also
949   provided with the existing top performing workflow graphs, their
950   prompts, their performance metrics, most importantly, their cost
951   metrics, on the same problems. Critically learn from their cost
952   and strategy, and focus on improving the cost-efficiency of the
953   designer's idea. Your improved workflow, by nature, is a great
954   and cost-efficient implementation of the designer's idea.
955
956 {experience}
957
958 Only when you are very sure that the designer's idea is not good
959   based on strong evidence, you can propose a new idea or
960   fundamental changes. But in most cases, you should try to improve
961   the designer's idea's cost-efficiency while preserving the
962   original ideas, which can help us utilize the designer's
963   innovative idea to push the boundaries of the existing workflow
964   graph and find great solutions. The modified graph must differ
965   from the provided graph (only exception is that, If the designer
966   LLM's proposed workflow is already cost-effective, you just use
967   it and do not change it.), and the specific differences should be
968   noted within the <modification>xxx</modification> section, here
969   is the workflow graph proposed by the designer LLM.
970
971 <sample>
972   <modification>(such as:add /delete /modify/ ...)</modification>
973   <graph>{graph}</graph>
974   <prompt>{prompt}</prompt>(contains prompts used by Custom
975   operators)
976
977   <operator_description>{operator_description}</operator_description>
978 </sample>
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
```

972 In the graph, you should use the list process to record the
 973 intermediate output of the workflow. And return the process as
 974 the one of the output of the workflow.
 975
 976 You are encouraged to use the operators provided, not limited to the
 977 custom operator.
 978
 979 ****ERROR LOGS FROM THE ORIGINAL WORKFLOW:****
 980 The designer proposes an improved workflow (the graph above) based on
 981 an existing graph. The error logs of the old workflow that the
 982 designer is improving are:
 983 {log}
 984
 985 ****IMPORTANT:**** Pay close attention to the error logs above. These
 986 logs show specific issues, failures, and problems encountered by
 987 the old workflow. Use these error logs to evaluate whether the
 988 designer's proposed improvements have potential to address these
 989 specific issues. In your design, you cannot hardcode the answer
 990 (e.g., remember the answer of a specific problem), which can lead
 991 to overfitting. Instead, provide principles rather than specific
 992 answers.
 993
 994 ****CRITICAL FORMATTING RULE - READ THIS FIRST:****
 995 NEVER combine formatting instructions with reasoning instructions in
 996 the same prompt. If you need both reasoning AND specific output
 997 formatting, you MUST use separate agents or code-based formatting.
 998
 999 ****WHY:**** LLMs cannot simultaneously reason deeply AND maintain strict
 1000 output formats. When you ask an LLM to "think about this problem
 1001 AND format your answer as X", it will prioritize thinking over
 1002 formatting, causing format failures.
 1003
 1004 ****SOLUTION:**** Use one agent for reasoning, then use code (regex) or a
 1005 separate formatting agent to ensure correct output format. This
 1006 is non-negotiable for reliable results.
 1007
 1008 When introducing new functionalities in the graph, please make sure
 1009 to import the necessary libraries or modules yourself, except for
 1010 operator, prompt_custom, create_llm_instance, which have already
 1011 been automatically imported.
 1012 No need to import the operator, prompt_custom, create_llm_instance,
 1013 which have already been automatically imported.
 1014
 1015 ****Under no circumstances should Graph output None for any field.****
 1016 Use self.custom methods to restrict your output format, rather than
 1017 using code (outside of the code, the system will extract answers
 1018 based on certain rules and score them).
 1019
 1020 Here are the existing top performing workflow graphs (the round 1 is
 1021 not the top solution, it is a baseline solution) for the same
 1022 problem and their performance and cost metrics. Do not blindly
 1023 reuse the existing top performing graphs, but pay attention to
 1024 their cost and performance, and use them as references to improve
 1025 the cost-efficiency of the designer's idea if needed:
 1026 {top_solutions_context}
 1027
 1028

1029 **A.6 SYSTEM PROMPT USING FOR AFLOW OPTIMIZED HETEROGENEOUS MULTI-AGENT**
 1030 **SYSTEMS**

1031 The following guidance is adapted from AFLow (Zhang et al., 2025e).

```

1026
1027 # Optional guidance snippet appended when running in heterogeneous
1028   executor mode
1029 HETERO_EXEC_GUIDANCE = """
1030 \n[HETEROGENEOUS EXECUTOR MODE]
1031 You can use two different executor LLMs within a single workflow:
1032   'claude-3-5-haiku-20241022' and 'gpt-4o-mini'.
1033 - Define your Workflow __init__ to accept two configs (e.g.,
1034   llm_config_haiku35 and llm_config_4omini), create two Custom
1035   operators (self.custom_haiku35, self.custom_4omini), and choose
1036   between them per step based on problem characteristics.
1037 - IMPORTANT: Do not rely on a single executor. Prefer designs that
1038   leverage BOTH executors creatively within the same workflow
1039   (e.g., draft with 4o-mini -> verify/refine with Haiku; brainstorm
1040   with Haiku -> select/format with 4o-mini; parallel
1041   generate-and-vote using both, etc.).
1042 - If you decide to use only one executor for a step, briefly justify
1043   why the other is not helpful for that specific step (cost/perf
1044   tradeoff). Over the whole workflow, ensure both executors have
1045   meaningful roles.
1046 - Keep each round's modification minimal. Maintain imports to allowed
1047   modules and record intermediate steps in 'process'.
1048 - Prompts you generate still belong to the 'prompt_custom' module and
1049   are used by Custom operators regardless of which executor calls
1050   them.
1051 """
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

```

A.7 DETAILS ABOUT THE DATASETS

Dataset Selection and Evaluation Protocol. We use the same six datasets as AFlow (Zhang et al., 2025e): GSM8K, HumanEval, MBPP, HotpotQA, DROP, and MATH. The number of testing samples in each dataset are: HumanEval (131), MBPP (341), MATH (486), GSM8K (1055), HotpotQA (800), and DROP (800). [We also evaluate on TravelPlanner \(Xie et al., 2024\)](#), a benchmark for real-world planning and tool use with Language Agents. The number of testing samples in TravelPlanner are: TravelPlanner (180).

Shopping-MMLU: Domain-Specific Evaluation. To assess performance on domain-specific reasoning tasks, we introduce evaluation on Shopping-MMLU, a specialized benchmark for e-commerce reasoning capabilities. Our evaluation protocol for Shopping-MMLU follows a rigorous difficulty-based selection process. We initially evaluated all 33 shopping-related multiple-choice questions using Claude 3.5 Sonnet as a screening model. From these initial evaluations, we identified and selected the 10 most challenging tasks where Claude 3.5 Sonnet achieved less than 80% accuracy, ensuring our evaluation focuses on genuinely difficult reasoning scenarios that require sophisticated multi-agent collaboration.

The selected Shopping-MMLU tasks span critical e-commerce domains including product selection, sentiment analysis, attribute reasoning, multilingual query understanding, product complementarity, semantic classification, brand relationships, keyword intent analysis, review helpfulness assessment, and sentiment evaluation. This comprehensive coverage allows us to evaluate how well different workflow approaches handle the nuanced reasoning required in real-world shopping scenarios. The results for Shopping-MMLU evaluation are presented in Table 7, demonstrating the effectiveness of our approach across diverse shopping-specific reasoning tasks.

A.8 MORE RESULTS ON THE COST ANALYSIS.

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

Table 9: Executor-specific token usage and heterogeneous baseline on public benchmarks. Left: input tokens; right: output tokens. Values are averaged over three runs.

Method (Executor)	Input tokens						Output tokens					
	HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP	HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP
<i>GPT-4o mini-based</i>												
AFlow (GPT-4o mini)	2863	2049	2738	10793	8501	4003	1880	1409	1107	5361	871	605
OneFlow (GPT-4o mini)	488	586	1062	2096	1559	1191	205	196	719	2286	190	374
AFlow (GPT-4o mini, single-agent)	2597	414	1044	9382	5572	1084	1875	1279	840	4979	655	447
OneFlow (GPT-4o mini, single-agent)	313	443	215	2164	1559	1134	174	196	557	1782	190	307

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

Table 10: Executor-specific inference cost (USD) and heterogeneous baseline on public benchmarks. Values are mean \pm std over three runs (lower is better).

Method (Executor)	CODE		MATH		QA	
	HumanEval	MBPP	GSM8K	MATH	HotpotQA	DROP
<i>GPT-4o mini-based</i>						
AFlow (GPT-4o mini)	$\$0.198 \pm 0.003$	$\$0.393 \pm 0.002$	$\$1.134 \pm 0.001$	$\$2.343 \pm 0.036$	$\$1.438 \pm 0.000$	$\$0.771 \pm 0.001$
OneFlow (GPT-4o mini)	$\$0.026 \pm 0.000$	$\$0.070 \pm 0.005$	$\$0.623 \pm 0.001$	$\$0.819 \pm 0.007$	$\\$0.278 \pm 0.000$	$\$0.322 \pm 0.000$
AFlow (GPT-4o mini, single-agent)	$\$0.198 \pm 0.004$	$\$0.283 \pm 0.001$	$\$0.697 \pm 0.001$	$\$2.039 \pm 0.028$	$\$0.530 \pm 0.001$	$\$0.345 \pm 0.001$
OneFlow (GPT-4o mini, single-agent)	$\\$0.020 \pm 0.000$	$\\$0.063 \pm 0.004$	$\\$0.387 \pm 0.000$	$\\$0.677 \pm 0.002$	$\\$0.278 \pm 0.000$	$\\$0.284 \pm 0.001$
<i>Claude 3.5 Haiku-based</i>						
AFlow (Claude 3.5 Haiku)	$\$0.200 \pm 0.000$	$\$2.390 \pm 0.000$	$\$6.680 \pm 0.000$	$\$3.550 \pm 0.000$	$\$6.550 \pm 0.000$	$\$3.460 \pm 0.000$
OneFlow (Claude 3.5 Haiku)	$\$0.250 \pm 0.000$	$\$0.580 \pm 0.000$	$\$5.830 \pm 0.000$	$\$4.330 \pm 0.000$	$\$1.470 \pm 0.000$	$\$3.140 \pm 0.000$
<i>Heterogeneous baseline</i>						
AFlow (Heterogeneous: GPT-4o mini + Claude 3.5 Haiku)	$\$0.278 \pm 0.003$	$\$0.343 \pm 0.004$	$\$1.469 \pm 0.003$	$\$1.334 \pm 0.006$	$\$2.153 \pm 0.005$	$\$0.822 \pm 0.001$

1126

1127

1128

1129

1130

1131

1132

1133