Inefficiencies of Meta Agents for Agent Design

Batu El[1] **Mert Yuksekgonul**[1] **James Zou**[1]

Abstract

Recent works began to automate the design of agentic systems using meta-agents that propose and iteratively refine new agent architectures. In this paper, we examine three key challenges in a common class of meta-agents. *First*, we investigate how a meta-agent learns across iterations and find that simply expanding the context with all previous agents, as proposed by previous works, performs worse than ignoring prior designs entirely. We show that the performance improves with an evolutionary approach. *Second*, although the meta-agent designs multiple agents during training, it typically commits to a single agent at test time. We find that the designed agents have low behavioral diversity, limiting the potential for their complementary use. *Third*, we assess when automated design is economically viable. We find that only in a few cases—specifically, two datasets—the overall cost of designing and deploying the agents is lower than that of human-designed agents when deployed on over 15,000 examples. In contrast, the performance gains for other datasets do not justify the design cost, regardless of scale.

1. Introduction

Agentic systems powered by language models demonstrated remarkable abilities to perform complex tasks and became a transformative force in many domains, including cutting-edge research and development (Swanson et al., 2024; Lu et al., 2024b; Yamada et al., 2025), financial services (Okpala et al., 2025; Xiao et al., 2025), and task automation (Fourney et al., 2024). Until recently, these systems were designed by researchers who built their domain knowledge into their agent architectures. However, a persistent trend in machine learning research, known as the Bitter Lesson (Sutton, 2019), suggests that hand-designed solutions are eventually replaced by solutions developed via scalable approaches that leverage *search* and *learning*. To this end, recent works have taken the first steps in the direction of automating the design of agentic systems (Hu et al., 2024; Li et al., 2024; Saad-Falcon et al., 2024; Niu et al., 2025; Nie et al., 2025; Shang et al., 2025; Wang et al., 2025; Ye et al., 2025; Zhang et al., 2025b;a). Our work focuses on a common class of meta-agents that follow the *sample–evaluate–iterate* pattern (see Figure 1, Algorithm 1) and highlights three challenges.

Meta Learning We begin by examining the assumption that the meta-agent effectively learns from previously discovered agents. Our analysis reveals that the meta-agent framework proposed by Hu et al. (2024) does not meaningfully leverage prior designs. In fact, it performs worse than a baseline that ignores prior designs entirely. In contrast, we demonstrate that an evolutionary context curation strategy, where the generation of the next agent is conditioned on the previous best-performing agents (parents), yields improved performance.

Diversity and Complementarity While the meta-agent generates a set of candidate agents, typically only one is deployed, neglecting potential synergies among them. If the designed agents were behaviorally diverse, where each specializes in particular types of queries, this would enable dynamic selection of the most suitable agent per query. However, we find that the designed agents often lack behavioral diversity, which is even more pronounced when evolutionary strategies are used.

Economic Viability For a meta-agent to be economically viable, the fixed cost of designing a new agent must be justified by corresponding improvements in performance. We formalize this trade-off by defining the total cost of a meta-designed agent as the sum of a fixed design cost and a per-example inference cost. This raises the key question: *How many test examples are needed before the cost per correct response becomes lower when using the designed agent?* In our experiments, we find this break-even point occurs at approximately $15{,}000$ examples for MMLU and DROP. In contrast, for other datasets, the performance gains do not justify the design cost, regardless of the scale of deployment.

[1] Stanford University. Correspondence to: Batu El <batuel@stanford.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

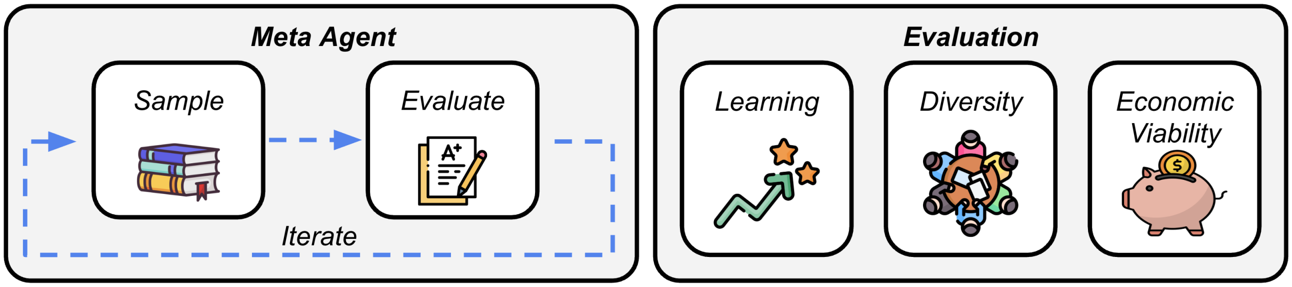

Figure 1. **Overview of the meta-agent framework.** The Meta-Agent iteratively samples and evaluates agents, refining its outputs through a feedback loop. We focus on three key dimensions: (1) learning from previously designed agents; (2) diversity and complementarity of generated agents; and (3) economic viability.

2. Related Works

Our primary reference is ADAS (Hu et al., 2024), which has introduced meta-agent search with the idea of searching for agents in the code space. MAS-GPT (Ye et al., 2025) and ScoreFlow (Wang et al., 2025) develop meta-agents by training a model to dynamically generate multi-agent systems for a given query. AgentSquare (Shang et al., 2025) and Archon (Saad-Falcon et al., 2024) explore modular agent architectures and use discrete module recombination to efficiently search design spaces. AutoFlow (Li et al., 2024), Weak-for-Strong (Nie et al., 2025), and ADAS (Hu et al., 2024) use a meta agent that follows the sample-evaluate-iterate paradigm (Algorithm 1). Other recent meta-agent approaches include Multi-agent Supernet (Zhang et al., 2025a), Flow (Niu et al., 2025), and AFlow (Zhang et al., 2025b). Erol et al. (2025) examined the cost of producing a correct response, which is directly relevant to our economic viability analysis.

Algorithm 1 Meta Agent: *Sample-Evaluate-Iterate*

```
$D_{\text{train}}$ # set of training examples
$F$ # initial agents library
$A = \{(f_{0_i}, s_{0_i}) \mid f_{0_i} \in F\}$ # archive
for $t$ in $[T]$ do
    $\hat{A} = \phi(A)$ # select current context
    $f_t \sim \Pi(\cdot \mid \hat{A})$ # sample, revise, debug
    $s_t = eval(f_t)$ # evaluate
    $A.append(f_t, s_t)$ # add to archive
    # iterate
end for
```

3. Setup

Following Hu et al. (2024), we define an agent as a computer program (Python function) that takes a question as input, makes language model calls to compute a response, and returns the result. Let f_i denote an agent and score $s_i = eval(f_i, D_{\text{train}}) \in \mathbb{R}^{N_{\text{train}}}$ be the evaluation vector containing the agent i's evaluation scores for each example in the training dataset D_{train}. The agent f_i is represented by code. The archive, A, is a set of discovered agents $\{f_i\}$ and their corresponding evaluations on the training set. We initialize the archive with the agents in the initial agents library, F, and their corresponding evaluations.[1] At each iteration, the meta-agent samples a new agent design using a language model, Π, conditioned on a curated subset of the current archive, $\hat{A}$. The function ϕ implements this curation step. The sampling step is followed by revisions to ensure proper formatting and debugging with execution feedback. Finally, the new agent, f_t, is added to the archive A.[2] Algorithm 1 outlines the design procedure. We experiment with three instantiations of context curation (ϕ):

Cumulative. ϕ_C is identity, and the generation of the next agent is conditioned on all the previously discovered architectures, as in Hu et al. (2024).

Parallel. ϕ_P maps any archive to only the subset that contains 7 agents in the initial library and corresponding evaluation scores. Hence, the meta-agent ignores the previously designed architectures, effectively parallel sampling the new agents.

Evolutionary. ϕ_E selects a subset of size 7 agents from A with the best evaluation scores (parents of the next agent). The generation of the next agent is conditioned on a higher quality subset of the previously discovered architectures at each iteration.

3.1. Tasks and Models

Closely following the prior work (Hu et al., 2024), we evaluate our agentic design setup on 1) mathematical reason-

[1] The content of F is discussed in Appendix A.1.

[2] Appendix A.2 elaborates on our experimental setup.

Dataset	Best Agent			Best-5 Avg.			Best-10 Avg.			Best-15 Avg.			Test Performance (Best Agent)			
	C	P	E	C	P	E	C	P	E	C	P	E	I	C	P	E
DROP	71.4	72.5	**74.4**	68.1	69.3	**71.5**	66.6	66.8	**69.7**	64.9	64.9	**68.2**	64.8	71.9	72.6	**73.2**
	(2.0)	(4.2)	(3.2)	(1.1)	(1.2)	(4.5)	(0.8)	(1.1)	(4.7)	(0.6)	(1.8)	(4.7)	(1.3)	(3.2)	(7.8)	(5.1)
MGSM	41.4	56.2	**56.5**	32.5	48.4	**50.4**	27.4	43.4	**46.0**	22.4	39.8	**42.7**	38.4	41.2	51.8	**53.5**
	(6.2)	(10.5)	(4.7)	(13.8)	(9.8)	(0.8)	(16.6)	(7.9)	(1.6)	(17.1)	(5.3)	(2.5)	(2.8)	(4.8)	(7.6)	(2.0)
MMLU	74.7	76.3	**76.6**	73.0	73.8	**74.8**	70.3	72.4	**73.7**	68.0	71.1	**72.7**	62.8	66.2	**67.8**	65.8
	(2.0)	(1.6)	(2.7)	(2.1)	(2.4)	(2.7)	(5.2)	(2.5)	(2.4)	(8.0)	(3.0)	(2.3)	(2.3)	(4.2)	(0.8)	(3.3)
GPQA	32.3	**35.2**	33.8	26.4	**32.2**	31.2	22.5	**30.4**	29.8	20.7	**29.1**	28.8	30.0	29.7	**31.3**	28.5
	(2.6)	(2.8)	(2.2)	(8.7)	(1.5)	(0.9)	(12.4)	(1.1)	(0.8)	(13.2)	(0.6)	(0.5)	(2.4)	(2.7)	(0.0)	(3.1)
Avg.	55.0	**60.0**	**60.3**	50.0	**55.9**	**57.0**	46.7	**53.2**	**54.8**	44.0	**51.2**	**53.1**	49.0	52.2	**55.9**	**55.2**

Table 1. **Meta-Agent Performance: Parallel context curation outperforms cumulative curation, while evolutionary approaches lead to further improvements.** Columns 1–12 report performance on D_{train} for: the single *Best Agent* (cols 1–3), and the averages of the top 5 (cols 4–6), top 10 (cols 7–9), and top 15 (cols 10–12) agents, evaluated under three context curation strategies: Cumulative (C), Parallel (P), and Evolutionary (E). Columns 13–16 show the D_{test} performance of the agent that achieves the highest score on D_{train}. I denotes the test performance of the best agent from the Initial library selected based on its training performance. Averaged across 3 runs.

ing abilities in a multi-lingual setting, MGSM, (Shi et al., 2022), 2) reading comprehension, DROP, (Dua et al., 2019), 3) multi-task problem solving, MMLU, (Hendrycks et al., 2021), and 4) graduate-level science questions, GPQA (Rein et al., 2023). From these datasets, we sample disjoint subsets D_{train} to compute s_i, and D_{test} to be used as held-out evaluation. The details of our experimental setup are explained in Appendix A.2. All the results we report are averaged across 3 runs.

4. Experiments

4.1. Learning

Table 1 compares three context curation strategies for meta-agent design. We find that *cumulative context curation does not outperform parallel context curation*, suggesting that ADAS-style meta-agents derive limited benefits from prior agent designs and perform worse than ignoring prior designs entirely.

In contrast, *evolutionary context curation improves performance*, yielding up to a $+10\%$ gain over cumulative context on MGSM. This suggests that selectively including high-quality prior designs in context enables more effective meta-learning.

4.2. Diversity and Complementarity

To investigate the potential synergies between the generated agents, we turn our attention to the behavioral diversity of the agent pool and analyze whether the agents have similar behavior on training examples. *How often the questions they get right overlap? Do they make the same mistakes?*

We analyze agent diversity by computing similarities between evaluation vectors. Let $s_i = \text{eval}(f_i, D_{\text{train}}) \in \mathbb{R}^{N_{\text{train}}}$ be the evaluation vector for agent f_i. Stacking s_i as rows, we obtain $\mathbf{S}$, which, in effect, represents embeddings of each agent from the perspective of the training questions (see Figure 6). We then compute the cosine similarity matrix $\mathbf{C}$, where the entry i, j corresponds to the cosine similarity $\langle s_i, s_j \rangle$ (see Figure 7). This pairwise similarity metric favors agents that succeed on the same examples. We show the histograms of pairwise similarities (entries of $\mathbf{C}$) in Figure 8 and the histograms for the average similarity of an agent to the rest of the agents (row averages of $\mathbf{C}$) in Figure 2.

Figure 2 shows the similarity distributions, with evolutionary context curation generally exhibiting higher similarity scores. We observe that *cumulative context curation yield lower similarity* overall compared to parallel and evolutionary context curation. Moreover, while *parallel and evolutionary context curation yield similar performance*, parallel context curation exhibits slightly lower similarity and produces more diverse agents. Notably, in GPQA, parallel context curation yields both better-performing and more diverse agents. Our analysis of coverage (Table 2)—the proportion of questions correctly answered at least by one of the designed agents—shows that *parallel context curation yields the highest coverage*, highlighting its effectiveness in promoting exploration.

	DROP	MGSM	MMLU	GPQA	Avg.
C	96.6	89.1	99.2	91.9	94.2
P	96.0	95.3	97.7	94.4	**95.9**
E	93.6	93.0	99.2	91.9	94.4

Table 2. Coverage. Proportion of questions correctly answered at least by one of the designed agents. The designed agents includes all 90 agents designed across 3 runs.

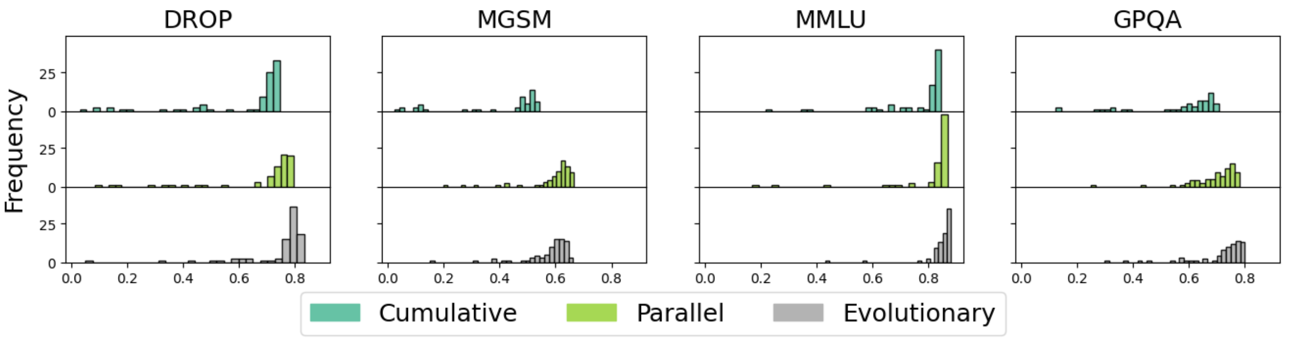

Figure 2. **Agent Diversity: Cumulative context curation yields lower overall similarity. Parallel context curation produces greater agent diversity compared to evolutionary curation, highlighting an exploration exploitation trade-off.** Histograms of agent similarities (row averages of **C**), excluding agents with zero performance (all-black rows of **S** in Figure 6, and corresponding dark blue rows and columns of **C** in Figure 7). Each subplot shows histograms of averaged similarity scores for each agent (x-axis) and their frequency (y-axis) across 3 runs.

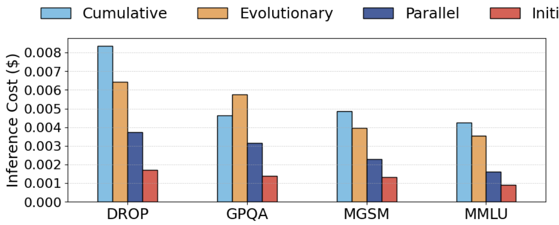

Figure 3. **Average inference cost per test query: C > E > P > I.** For agents in the initial library F (Initial), agents designed by meta agent with ϕ_C (Cumulative), agents designed by meta agent with ϕ_P (Parallel), agents designed by meta agent with ϕ_E (Evolutionary). Averaged across all agents from 3 runs.

4.3. Economic Viability

In Figure 3, we observe that the agents designed using ϕ_C have the highest average inference costs, followed by those designed using evolutionary context, ϕ_E. Among the meta agents, the one that uses parallel context curation produces the least costly agents on average, a trend also observed among the best-performing agents (Figure 5). However, agents designed by the meta agent still remain more costly than those in the initial library.

To identify the point where the cost per correct response of a designed agent becomes lower than the agents in the initial agents library, we combine the inference cost of the best agent (Figure 5) with the fixed cost of agent design. The fixed design cost, C_0, includes the total cost of all the sampling step (Algorithm 1, line 6; Figure 13) and evaluation costs to compute s_i (Algorithm 1, line 7). The total cost of an agent is the sum of C_0 and a per-example inference cost, C_{j}:$C_0 + n \cdot C_{\mathrm{j}}$.

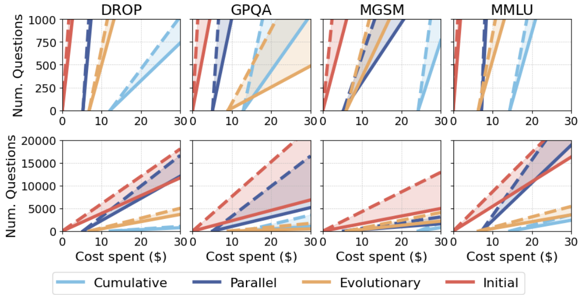

Figure 4. **Cost Efficiency: Highest performing agent from the initial library generates the outputs with same total performance at lower cost.** Number of questions solved (solid lines) and attempted (dashed lines) versus cost spent for agents with best training set performance. The x-intercept indicates the fixed cost C_0 (0 for agents in initial library); the slope beyond reflects variable cost per attempt or per solution.

In Figure 4, the intersection of the red solid line with another solid line marks the break-even point, where deploying the meta-agent lowers the cost per correct response. This occurs at approximately $n = 15{,}000$ examples for DROP and MMLU with parallel context curation. In contrast, for other datasets and context curation methods, performance gains do not justify the associated costs at any scale.

5. Conclusion

Our analysis highlights key trade-offs between (1) final performance and behavioral diversity and (2) performance relative to cost. Evolutionary context curation boosts performance but reduces diversity. While meta-agent-driven design can produce cost-effective agents in some cases, the performance gains rarely justify the increased design and inference costs, even at scale.

Impact Statement

For meta-agents, the unchecked generation and execution of complex systems may present safety risks. Such systems are difficult to audit or control prior to deployment within automated design loops.

References

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate, 2023. URL https://arxiv.org/abs/2305.14325.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Erol, M. H., El, B., Suzgun, M., Yuksekgonul, M., and Zou, J. Cost-of-pass: An economic framework for evaluating language models, 2025. URL https://arxiv.org/abs/2504.13359.

Fernando, C., Banarse, D., Michalewski, H., Osindero, S., and Rocktäschel, T. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023. URL https://arxiv.org/abs/2309.16797.

Fourney, A., Bansal, G., Mozannar, H., Tan, C., Salinas, E., Erkang, Zhu, Niedtner, F., Proebsting, G., Bassman, G., Gerrits, J., Alber, J., Chang, P., Loynd, R., West, R., Dibia, V., Awadallah, A., Kamar, E., Hosn, R., and Amershi, S. Magentic-one: A generalist multi-agent system for solving complex tasks, 2024. URL https://arxiv.org/abs/2411.04468.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Hu, S., Lu, C., and Clune, J. Automated design of agentic systems, 2024. URL https://arxiv.org/abs/2408.08435.

Li, Z., Xu, S., Mei, K., Hua, W., Rama, B., Raheja, O., Wang, H., Zhu, H., and Zhang, Y. Autoflow: Automated workflow generation for large language model agents, 2024. URL https://arxiv.org/abs/2407.12821.

Lu, C., Hu, S., and Clune, J. Intelligent go-explore: Standing on the shoulders of giant foundation models, 2024a. URL https://arxiv.org/abs/2405.15143.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The ai scientist: Towards fully automated open-ended scientific discovery, 2024b. URL https://arxiv.org/abs/2408.06292.

Lu, P., Chen, B., Liu, S., Thapa, R., Boen, J., and Zou, J. Octotools: An agentic framework with extensible tools for complex reasoning, 2025. URL https://arxiv.org/abs/2502.11271.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Nie, F., Feng, L., Ye, H., Liang, W., Lu, P., Yao, H., Alahi, A., and Zou, J. Weak-for-strong: Training weak meta-agent to harness strong executors, 2025. URL https://arxiv.org/abs/2504.04785.

Niu, B., Song, Y., Lian, K., Shen, Y., Yao, Y., Zhang, K., and Liu, T. Flow: Modularized agentic workflow automation, 2025. URL https://arxiv.org/abs/2501.07834.

Okpala, I., Golgoon, A., and Kannan, A. R. Agentic ai systems applied to tasks in financial services: Modeling and model risk management crews, 2025. URL https://arxiv.org/abs/2502.05439.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Saad-Falcon, J., Lafuente, A. G., Natarajan, S., Maru, N., Todorov, H., Guha, E., Buchanan, E. K., Chen, M., Guha, N., Ré, C., and Mirhoseini, A. Archon: An architecture search framework for inference-time techniques, 2024. URL https://arxiv.org/abs/2409.15254.

Shang, Y., Li, Y., Zhao, K., Ma, L., Liu, J., Xu, F., and Li, Y. Agentsquare: Automatic llm agent search in modular design space, 2025. URL https://arxiv.org/abs/2410.06153.

Shi, F., Suzgun, M., Freitag, M., Wang, X., Srivats, S., Vosoughi, S., Chung, H. W., Tay, Y., Ruder, S., Zhou, D., Das, D., and Wei, J. Language models are multilingual chain-of-thought reasoners, 2022. URL https://arxiv.org/abs/2210.03057.

Singh, H., Das, R. J., Han, M., Nakov, P., and Laptev, I. Malmm: Multi-agent large language models for zero-shot robotics manipulation, 2024. URL https://arxiv.org/abs/2411.17636.

Su, H., Chen, R., Tang, S., Yin, Z., Zheng, X., Li, J., Qi, B., Wu, Q., Li, H., Ouyang, W., Torr, P., Zhou, B., and Dong, N. Many heads are better than one: Improved scientific idea generation by a llm-based multi-agent system, 2025. URL https://arxiv.org/abs/2410.09403.

Sutton, R. The bitter lesson, 2019. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html. Accessed: 2025-01-04.

Swanson, K., Wu, W., Bulaong, N. L., Pak, J. E., and Zou, J. The virtual lab: Ai agents design new sars-cov-2 nanobodies with experimental validation. *bioRxiv*, 2024. doi: 10.1101/2024.11.11.623004. URL https://www.biorxiv.org/content/early/2024/11/12/2024.11.11.623004.

Wang, Y., Yang, L., Li, G., Wang, M., and Aragam, B. Scoreflow: Mastering llm agent workflows via score-based preference optimization, 2025. URL https://arxiv.org/abs/2502.04306.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Xiao, Y., Sun, E., Luo, D., and Wang, W. Tradingagents: Multi-agents llm financial trading framework, 2025. URL https://arxiv.org/abs/2412.20138.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J., Clune, J., and Ha, D. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066.

Ye, R., Tang, S., Ge, R., Du, Y., Yin, Z., Chen, S., and Shao, J. Mas-gpt: Training llms to build llm-based multi-agent systems, 2025. URL https://arxiv.org/abs/2503.03686.

Yin, X., Wang, X., Pan, L., Wan, X., and Wang, W. Y. Gödel agent: A self-referential agent framework for recursive self-improvement, 2025. URL https://arxiv.org/abs/2410.04444.

Zelikman, E., Lorch, E., Mackey, L., and Kalai, A. T. Self-taught optimizer (stop): Recursively self-improving code generation, 2024. URL https://arxiv.org/abs/2310.02304.

Zhang, G., Niu, L., Fang, J., Wang, K., Bai, L., and Wang, X. Multi-agent architecture search via agentic supernet, 2025a. URL https://arxiv.org/abs/2502.04180.

Zhang, J., Xiang, J., Yu, Z., Teng, F., Chen, X., Chen, J., Zhuge, M., Cheng, X., Hong, S., Wang, J., Zheng, B., Liu, B., Luo, Y., and Wu, C. Aflow: Automating agentic workflow generation, 2025b. URL https://arxiv.org/abs/2410.10762.

Zhou, W., Ou, Y., Ding, S., Li, L., Wu, J., Wang, T., Chen, J., Wang, S., Xu, X., Zhang, N., Chen, H., and Jiang, Y. E. Symbolic learning enables self-evolving agents, 2024. URL https://arxiv.org/abs/2406.18532.

A. Experimental Setup Details

A.1. Initial Agents Library

Our initial agent library, F, consists of the following methods: (1) **Chain-of-Thought** (Wei et al., 2023), which prompts the language model to output its reasoning before arriving at an answer; (2) **Majority Voting**, which selects the consensus response from multiple generated answers; (3) **Refinement from Feedback** (Madaan et al., 2023), where the model iteratively improves its answer based on self-feedback; (4) **LLM-Debate** (Du et al., 2023), where multiple language model instances are prompted to debate with each other; (5) **Quality-Diversity** (Lu et al., 2024a), which generates and ensembles diverse responses; (6) **Routing**, which directs tasks to the most appropriate language model instances prompted to behave like an expert of a subject; and (7) **Stepping-back** (Hu et al., 2024), which encourages the model to first reflect on relevant scientific principles before answering. This is consistent with the setup in Hu et al. (2024).

A.2. Experimental Setup

Number of Iterations. In all our experiments, we use $T = 30$.

Dataset Size. For each of our MGSM, MMLU, DROP datasets, we select 128 examples from our dataset as training examples, denoted as D_{train}, and 200 examples as test examples, denoted as D_{test}. For GPQA, we use 32 samples as training examples and the remaining 160 samples as test examples. To reduce the variance during training, we use each training example from GPQA 5 times and compute scores using $5 \times 32 = 160$ evaluations. Performance is measured using F1-score for DROP and accuracy for the other datasets.

Models. In our experiments, we use `gpt-3.5` as the engine of the `LanguageModel` class. We use a larger, more powerful model, `gpt-4o`, as the engine of the meta-agent. This is consistent with the setup in Hu et al. (2024).

B. Other Related Works

Agentic Systems Agentic systems have demonstrated remarkable success across a range of domains. Several agentic systems have advanced scientific automation, including frameworks for end-to-end research (Lu et al., 2024b), autonomous paper writing (Yamada et al., 2025), nanobody design in a virtual lab (Swanson et al., 2024), and multi-agent ideation (Su et al., 2025). Beyond research, agentic systems have demonstrated effectiveness in complex operational contexts, including generalist problem-solving (Fourney et al., 2024; Lu et al., 2025), financial modeling and trading (Okpala et al., 2025; Xiao et al., 2025), and robotics manipulation (Singh et al., 2024).

Recursive Self-Improvement STOP (Zelikman et al., 2024), Promptbreeder (Fernando et al., 2023), Gödel Agent (Yin et al., 2025), and Zhou et al. (2024) implement recursive self-improvement by enabling agents to iteratively refine their own prompts, code, or internal reasoning logic.

C. Limitations

Scope Our work focuses on a class of meta-agent approaches that follow the sample–evaluate–iterate pattern. While restricting our scope to this setup enables us to highlight general patterns, our findings may not apply directly to the broader space of possible meta-agent paradigms.

Evaluation We evaluate performance primarily in terms of accuracy and F-1 scores. Our findings may not directly translate to domains where consistency is critical, or where different utility metrics are more appropriate.

Economic Viability Our analysis of economic viability is most suited for domains with strong verifiers as it emphasizes the cost of sampling a correct or high-performing answer. Other formulations may be better suited for different applications.

Similarity Computation Cosine similarity favors alignment between agents that succeed on the same examples. The metric reaches its maximum (1) when agents can solve the same set of questions. However, favoring alignment introduces an overall bias toward high-performing agents. Due to this bias, high-performing agents appear more similar, whereas agents that fail consistently appear orthogonal. As a robustness check, we also computed Hamming distances between binary score vectors and observed similar trends (Figure 9, 10).

Meta Evaluation Meta-agent evaluations involve multiple sources of stochasticity, including (1) LM output randomness, (2) error propagation in chained reasoning inside agents, (3) meta-agent sampling variability, (4) meta trajectory-level divergence due to different sampled agents, and (5) stochasticity in training evaluation results for the designed agents, which can then lead the trajectories in different directions. Robust evaluation thus requires multiple trajectory samples for reliable conclusions. Due to the extensive costs of larger-scale evaluations, the results we present are averaged across 3 runs.

D. Additional Results

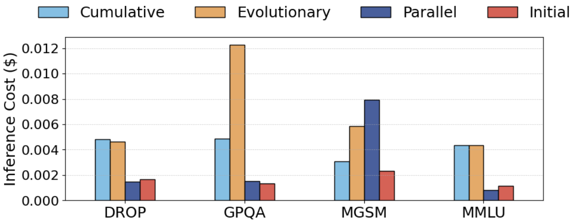

Figure 5. **Average inference cost per test query of the best agents.** For best agent in the initial library F (Initial, see Appendix A.1), best agent designed by meta agent with ϕ_C (Cumulative), best agents designed by meta agent with ϕ_P (Parallel), best agent designed by meta agent with ϕ_E (Evolutionary). Averaged across the single best agents from 3 runs. Best agent is selected based on the highest training performance.

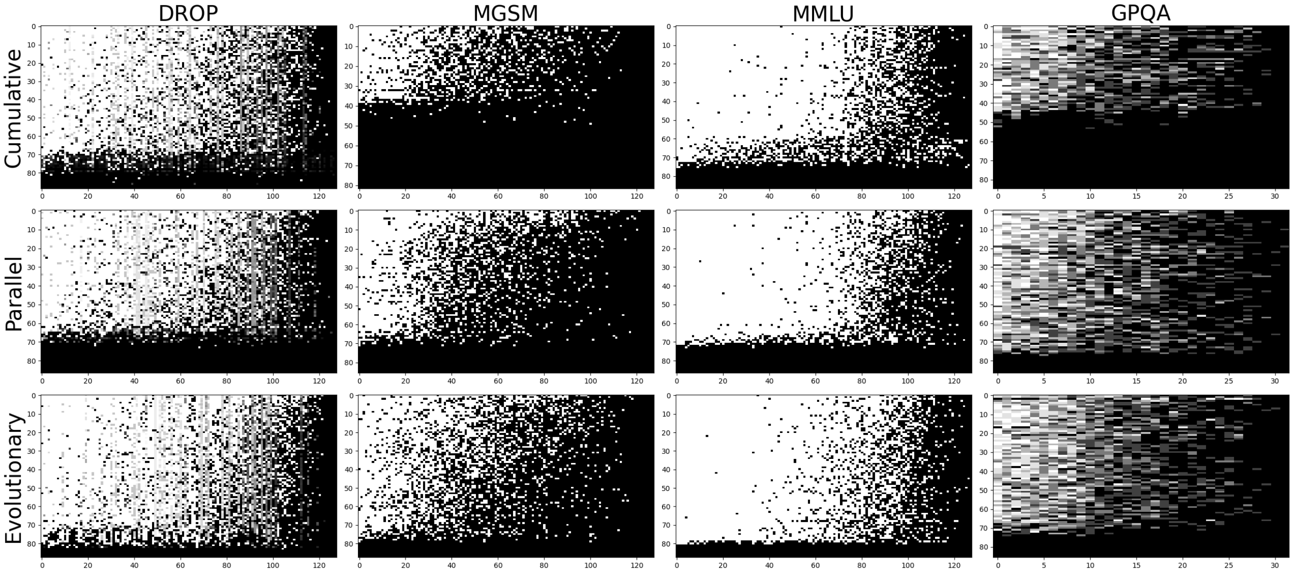

Figure 6. Score matrix $\mathbf{S}$, where each row corresponds to an agent and each column to a dataset example. A cell is white if the agent answers correctly and black otherwise. For DROP, gray indicates intermediate F1 scores; for GPQA, gray denotes partial correctness across repeated attempts. The normalized rows, s_i, serve as agent embeddings, capturing performance across training questions.

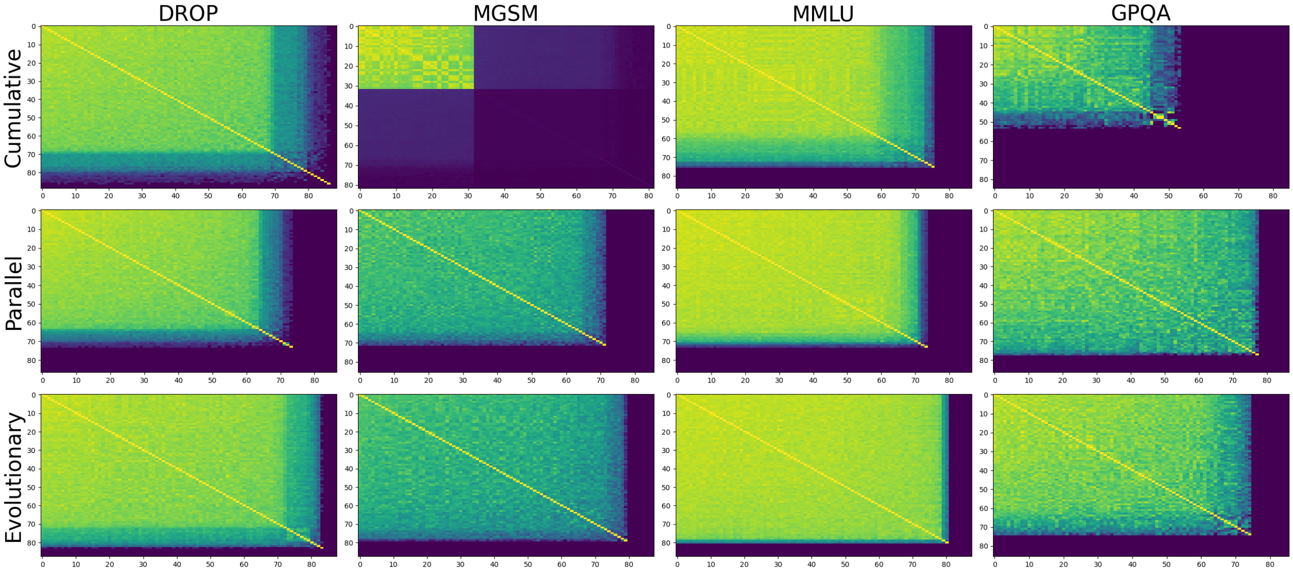

Figure 7. Cosine similarity matrix $\mathbf{C}$, with agents reordered by descending average similarity to all other agents.

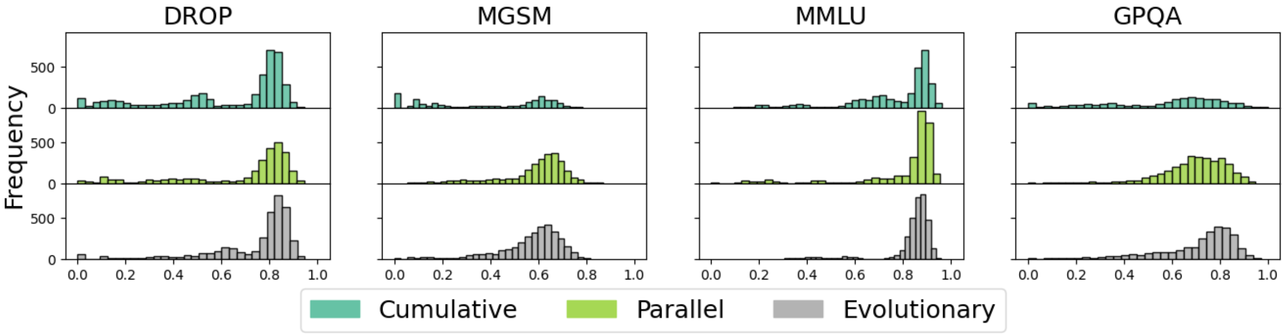

Figure 8. Histograms of agent similarities (entries of $\mathbf{C}$), excluding agents with zero performance (all black rows of $\mathbf{S}$ in Figure 6, and corresponding dark blue rows and columns of $\mathbf{C}$ in Figure 7). Only the upper triangular entries of $\mathbf{C}$ (excluding the diagonal) are used, as $\mathbf{C}$ is symmetric. Each subplot shows histograms of similarity scores (x-axis) and their frequency (y-axis).

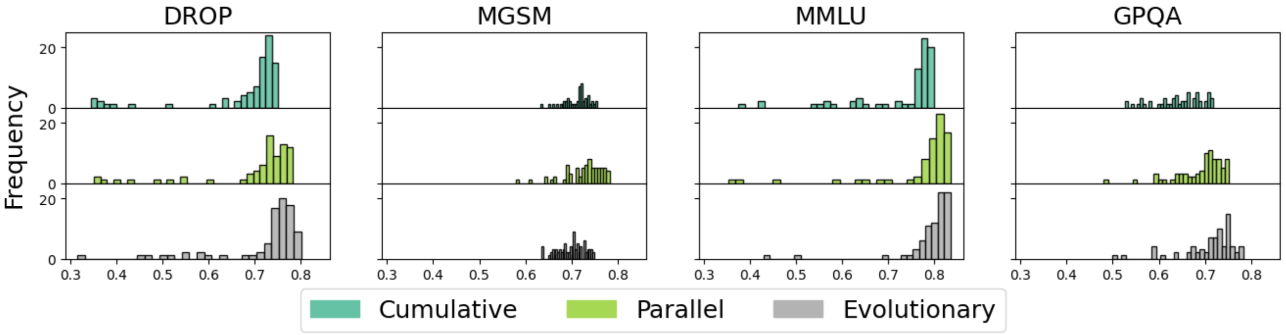

Figure 9. Figure 2 with (1 - Hamming distance) as the similarity metric. All nonzero entries of S are set to 1.

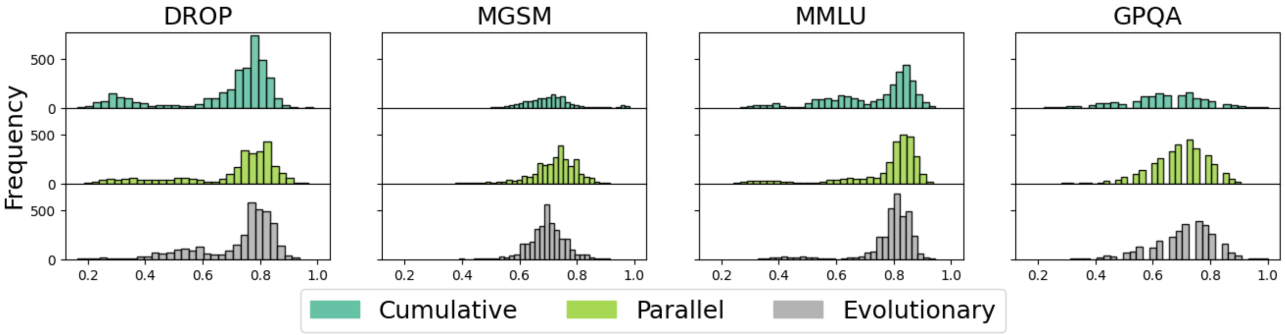

Figure 10. Figure 8 with (1 - Hamming distance) as the similarity metric. All nonzero entries of S are set to 1.

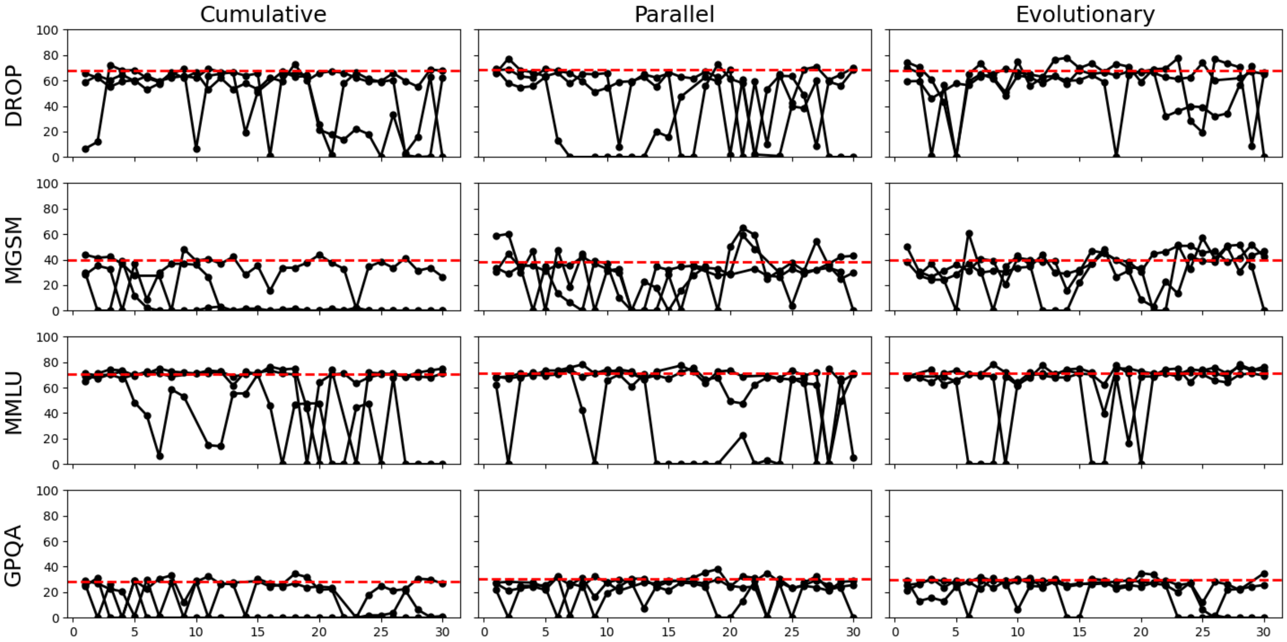

Figure 11. Training performance of designed agents across iterations. The dotted red line shows the performance of the best agent from the initial library.

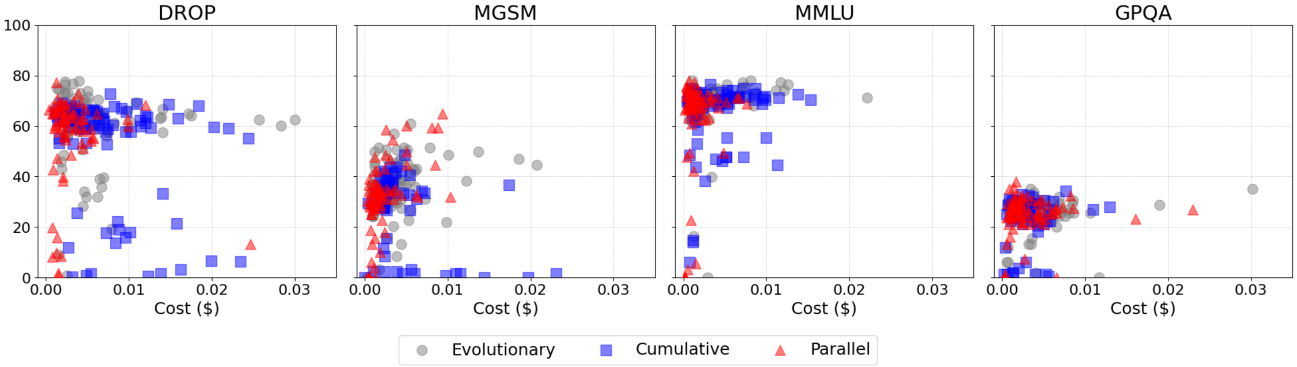

Figure 12. Performance (y axis) versus the inference cost (x-axis) of the designed agents.

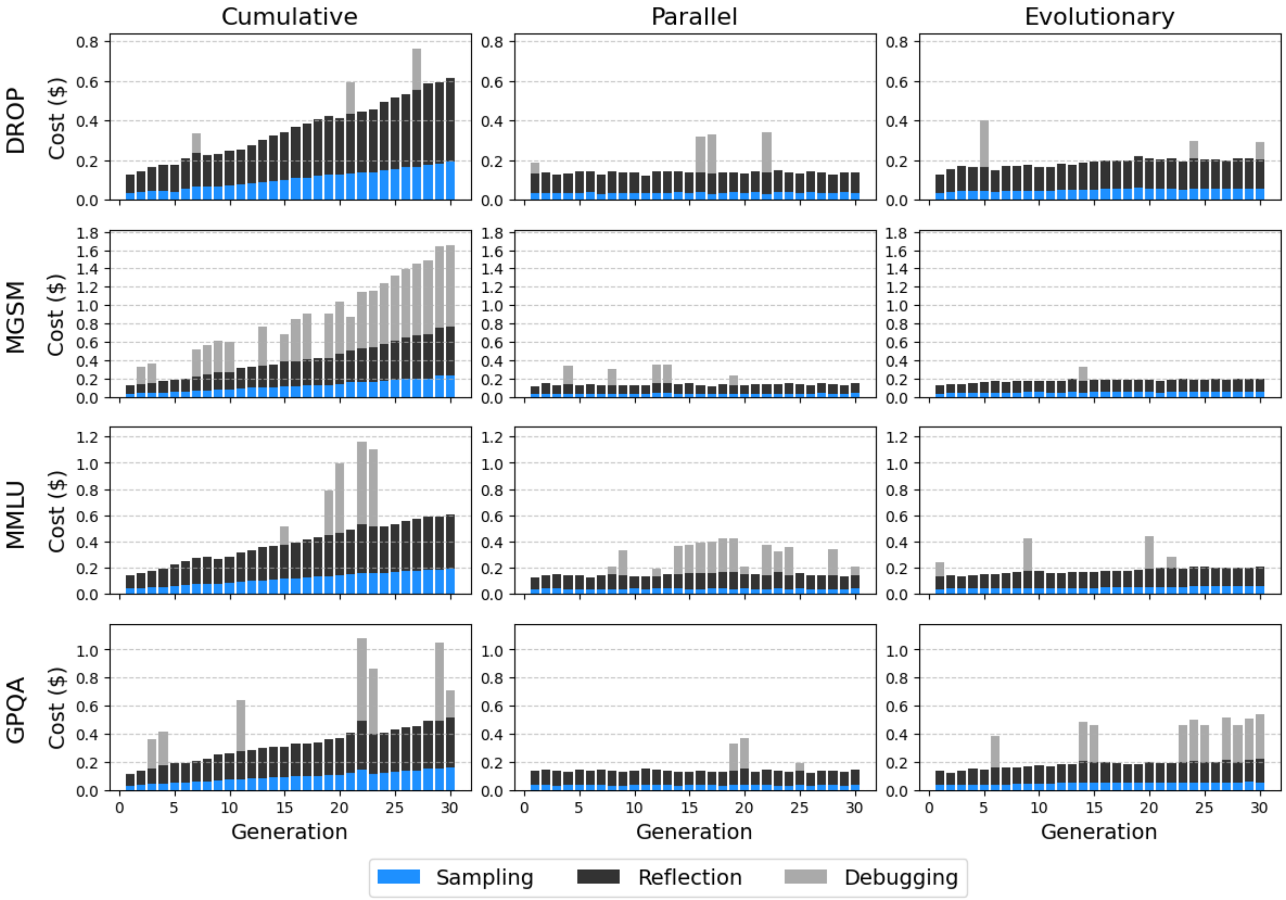

Figure 13. Design cost of the next agent across iterations. While costs remain stable with Parallel and Evolutionary context curation, they increase linearly with increasing context length in Cumulative context curation.