AGENTPSO: EVOLVING AGENT REASONING SKILL VIA MULTI-AGENT PARTICLE SWARM OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent reasoning has shown promise for improving the problem-solving ability of large language models by allowing multiple agents to explore diverse reasoning paths. However, most existing multi-agent methods rely on inference-time debate or aggregation, which can be vulnerable to incorrect peer influence and biased consensus. Moreover, the agents' reasoning procedures remain static, as they do not accumulate reusable behaviors across tasks. In this paper, we introduce **AgentPSO**, a particle-swarm-inspired framework for evolving multi-agent reasoning skills. AgentPSO treats each agent as a particle-like reasoner whose state is a natural-language skill and whose velocity is a semantic update direction, iteratively guiding agents toward higher-performing skill configurations. Across training iterations, each agent updates its skill by combining its previous velocity, personal-best skill, global-best skill, and a self-reflective direction derived from peer reasoning trajectories. This enables agents to learn reusable reasoning behaviors by drawing on their own experience and on the strongest skills found by the population, without updating the parameters of the backbone language model. Experiments on mathematical and general reasoning benchmarks show that AgentPSO improves over static single-agent skills and test-time-only multi-agent reasoning baselines. The evolved skills further transfer across benchmarks and to another backbone model, suggesting that AgentPSO captures reusable reasoning procedures rather than merely optimizing benchmark-specific prompts.

1 INTRODUCTION

"*Groups are remarkably intelligent, and are often smarter than the smartest people in them.*"
— James Surowiecki, *The Wisdom of Crowds* Surowiecki (2004)

Large language models (LLMs) have shown remarkable progress in solving complex reasoning problems OpenAI (2026); Achiam et al. (2023); Anthropic (2026), yet their performance remains highly sensitive to how they are prompted and what reasoning skills they are instructed to use Wei et al. (2022); Zheng et al. (2024); Yao et al. (2023); Wang et al. (2023); Madaan et al. (2023); Zhou et al. (2023a). As manually identifying optimal prompts and agent skills is labor-intensive, recent work has therefore explored automatic refinement of prompts, policies, and reasoning behaviors Prasad et al. (2022); Zhou et al. (2023b); Yang et al. (2024); Khattab et al. (2024); Azim et al. (2025); Agrawal et al. (2026); Zhou et al. (2026). However, most prompt optimization and self-evolution methods focus on improving a single agent, prompt, or pipeline, which can limit exploratory diversity and lead to local optima Huang et al. (2023).

This motivates the use of multi-agent systems, where multiple agents provide diverse reasoning perspectives beyond the limitations of a single agent's self-contained evolution. By critiquing one another and exposing alternative solution paths, multi-agent systems enable broader exploration of the reasoning space Du et al. (2023); Lin & Hooi (2025); Liu et al. (2025a; 2024). However, existing multi-agent paradigms largely rely on inference-time collaboration, such as multi-round debate, which can be vulnerable to incorrect peer influence, biased consensus, and substantial computational overhead Kaesberg et al. (2025); Cui et al. (2026).

Crucially, while these methods improve the outcomes of the current discussion, they do not evolve the agents themselves or enable them to internalize reusable reasoning skills from prior interactions. Although memory-based approaches store and retrieve past lessons Liu et al. (2025b); Ling et al.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

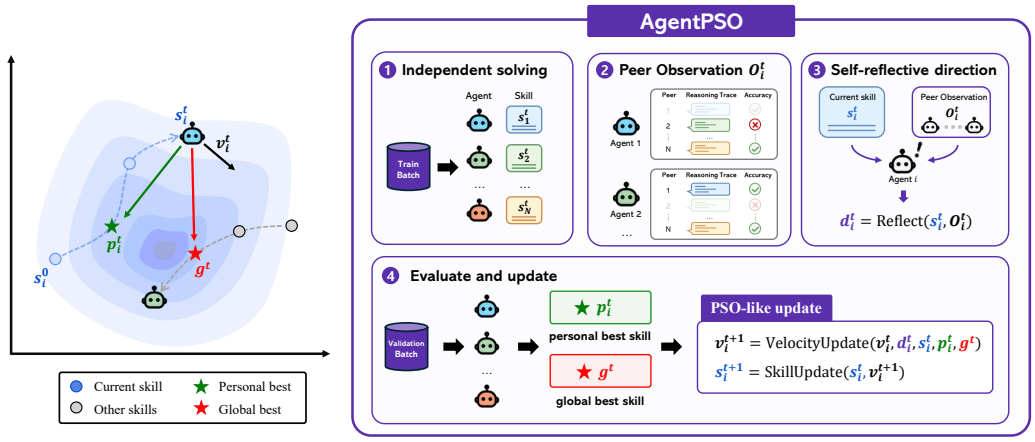


Figure 1: **Overview of AgentPSO.** Each agent independently solves the training batch with its current skill, producing answers and reasoning traces. Peer observations summarize other agents' reasoning traces and correctness, from which each agent derives a self-reflective direction for skill improvement. This direction is combined with guidance from the personal-best and global-best skills through a PSO-like update.

(2025), they mainly reuse externalized experiences at inference time instead of updating the agents' underlying reasoning skills. As a result, their ability to transfer general reasoning behaviors across benchmarks remains limited.

We therefore ask: *Can multi-agent systems iteratively evolve their underlying reasoning skills through collective interaction, rather than merely refining the output for a single instance?* We propose AgentPSO, a population-based framework for optimizing agent skills, inspired by the high-level structure of Particle Swarm Optimization (PSO) Kennedy & Eberhart (1995). In our framework, each agent is treated as a particle, and its velocity v captures the direction of skill refinement. At each iteration, agents update their skills by leveraging both their personal-best skill p and the global-best skill g. Through this population-guided update, AgentPSO exploits strong skills discovered by the swarm while maintaining diverse, agent-specific improvement trajectories.

However, directly applying the standard PSO rule to agent evolution is insufficient, as agents may move toward high-performing skill states without acquiring the reasoning principles that make them effective Freitas et al. (2020); Gad (2022); Zhang et al. (2019). AgentPSO therefore introduces a self-reflective direction, which allows each agent to analyze peer reasoning trajectories and extract reusable lessons instead of simply copying stronger skills. This reflection mechanism allows agents to encode reusable procedural reasoning patterns into their externalized skill representations. Over iterations, the initially diverse agents evolve into a collectively stronger population. At test time, the evolved agents solve problems independently and aggregate their answers through majority voting, achieving strong performance without costly multi-round interaction.

We evaluate AgentPSO on mathematical and general reasoning benchmarks. Our experiments show that AgentPSO outperforms single-agent prompting methods and debate-based multi-agent baselines. Further analyses show that the evolved skills transfer across benchmarks, improve progressively over iterations, and benefit substantially from the self-reflective direction.

Our contributions are summarized as follows:

- We introduce **AgentPSO**, a PSO-inspired framework that evolves multi-agent reasoning skills by updating agents through personal-best, global-best, and self-reflective directions.
- We demonstrate that AgentPSO yields persistent evolution of agent capabilities, achieving stronger performance than single-agent and debate-based multi-agent baselines without requiring costly test-time debate.
- We demonstrate that evolved skills transfer across benchmarks, indicating that AgentPSO learns general reasoning behaviors rather than merely memorizing task-specific knowledge.

2 Related Work

Multi-Agent Collaboration Approaches. Individual language models are often limited in their ability to explore diverse reasoning trajectories Huang et al. (2023), motivating multi-agent collaboration in which agents exchange information and refine their solutions collectively. Prior work has shown that multi-agent debate can improve reasoning by enabling agents to critique one another and expand the space of possible solutions Du et al. (2023); Liu et al. (2025a). Later studies improve reliability through confidence-aware interaction Lin & Hooi (2025), multidimensional assessment Feng et al. (2025), cross-agent reward signals Xue et al. (2025), and tournament-style comparison Fu et al. (2026). Other approaches augment collaboration with external memory, allowing agents to retrieve lessons from prior debates for future tasks Liu et al. (2024; 2025b). However, these methods remain vulnerable to negative peer influence Kaesberg et al. (2025); Cui et al. (2026); Weng et al. (2025), require costly inference-time collaboration, and do not directly improve agents' underlying capabilities across tasks. By contrast, we formulate multi-agent collaboration as agent skill evolution, where agents improve by learning from each other's reasoning trajectories.

Self-Evolution in Language Models. Recent research has increasingly aimed to make language models self-evolving by iteratively adapting their behavior to better fit a target task ang Gao et al. (2026). Early efforts focus on prompting, where task-relevant reasoning depends heavily on the prompts provided to the model Wei et al. (2022); Madaan et al. (2023); Khot et al. (2023); Shinn et al. (2023). Subsequent works therefore explore prompt optimization to automatically discover more effective prompts Prasad et al. (2022); Zhou et al. (2023b); Yang et al. (2024); Khattab et al. (2024); Azim et al. (2025); Fernando et al. (2024); Agrawal et al. (2026). More recent studies further extend this line of research from prompt optimization to the iterative refinement of reasoning structures, workflows, agent architectures, and externalized skills through feedback and self-improvement Zhou et al. (2024); Su et al. (2025); Hu et al. (2024); Zhang et al. (2026); Xue et al. (2025); Almansoori et al. (2025); Wei et al. (2025); Wang et al. (2025b); Zhou et al. (2026). Our method follows this direction, but focuses on population-level multi-agent skill evolution rather than optimizing a single component, such as a prompt, agent, or pipeline. Specifically, each agent updates its own skill based on its current state, a self-reflective update direction, its personal-best skill, and the global-best skill discovered across the population.

Particle Swarm Optimization. We interpret multi-agent skill evolution through the lens of Particle Swarm Optimization (PSO) Kennedy & Eberhart (1995), in which each particle is updated based on its current state, its personal best, and the swarm's global best. PSO has been widely applied to diverse optimization problems, including in language model optimization Shinohara et al. (2025); Gad (2022). We extend this perspective to multi-agent evolution by treating each agent as a particle-like reasoner with an evolving skill state. Each agent is thus guided by both its own personal-best skill and the strongest skill discovered across the population, enabling improvement at both the individual and collective levels. SwarmAgentic Zhang et al. (2025) also applies swarm intelligence to automated agentic system generation by jointly optimizing agent functionality and collaboration structures. However, standard PSO alone can be limited on complex search problems, where effective exploration remains difficult Freitas et al. (2020); Zhang et al. (2019). To address this limitation, our method extends the PSO-style update with a self-reflective *semantic* direction term by leveraging the capabilities of LLMs, encouraging broader exploration while steering agents toward better skill configurations.

3 Problem Setup

We consider a population of N-LLM agents $A = \{A_1, A_2, \ldots, A_N\}$ that jointly evolve their reasoning skills over problem-solving iterations. At iteration t, each agent A_i has a *reasoning skill* state $s_i^t \in \mathcal{S}$, where $\mathcal{S}$ denotes the space of valid skill states. Here, the reasoning skill s_i^t specifies reusable procedural behaviors—such as how the agent decomposes a problem, selects a solution strategy, verifies intermediate steps, and constructs the final answer. Accordingly, skill evolution refers to the iterative refinement of these externalized reasoning procedures. The agents are initialized with diverse skill instructions, allowing the population to start from multiple reasoning strategies rather than a single shared prompt.

Table 1: **Conceptual mapping between standard PSO and AgentPSO.** AgentPSO maps high-level PSO concepts to natural-language skill optimization, replacing numerical vector arithmetic with LLM-based semantic update operators.

	Standard PSO	**AgentPSO**
State	Continuous position x_i^t	Skill state s_i^t
Direction	Vector velocity v_i^t	Semantic velocity v_i^t
Velocity update	$\omega v_i^t + c_1 r_1 (p_i^t - x_i^t) + c_2 r_2 (g^t - x_i^t)$	$\mathrm{VelocityUpdate}(v_i^t, d_i^t, s_i^t, p_i^t, g^t)$
State update	$x_i^t + v_i^{t+1}$	$\mathrm{SkillUpdate}(s_i^t, v_i^{t+1})$

We define $F(s)$ as the validation accuracy obtained when an agent solves a validation batch using skill s. Our objective is to obtain a population of high-performing skill states $\{s_i^*\}_{i=1}^N$ that improve both individual agent performance and collective prediction. We optimize only the externalized skill representation, without updating the internal parameters of the backbone LLM. Because the optimized object is a reusable instruction rather than a task-specific answer, the resulting skills are expected to improve performance on unseen problems. During optimization, training batches are used to generate peer observations and update directions, whereas validation batches are used to select personal-best and global-best skills.

Our goal is not to optimize each skill independently. If each agent only updates its own instruction, the search remains limited to that agent's local trajectory. Instead, we use the population as a source of comparative evidence. When multiple agents solve the same batch, their reasoning traces reveal which strategies succeed, which fail, and which reasoning behaviors are useful across problems. Thus, skill evolution is driven not only by an agent's own experience, but also by behavioral evidence extracted from peer trajectories.

To perform this population-based skill search, we adopt a swarm-inspired formulation. Each agent maintains its current skill, its own search history, the personal-best skill it has found so far, and the global-best skill found by the population. In addition, each agent uses peer reasoning trajectories to infer how its current skill should be revised. We treat each agent A_i as a particle whose position is its current skill state s_i^t. Unlike standard particle-based optimization, however, this position is not a numerical vector but a natural-language instruction.

4 AGENTPSO

AgentPSO is a PSO-inspired framework for multi-agent reasoning skill evolution. Rather than directly instantiating numerical PSO updates, AgentPSO reformulates the personal-best and population-best mechanisms as semantic skill-revision operators, enabling natural-language skill optimization. AgentPSO adapts this structure to natural-language skill optimization by treating each skill instruction as a particle state and each textual revision direction as a semantic velocity. Table 1 summarizes this correspondence.

Each training iteration follows four steps: (i) All agents solve the same training batch independently using their current skills. (ii) Each agent observes the answer correctness and the reasoning traces produced by the population and derives a self-reflective update direction. (iii) Each agent combines this direction with its previous semantic velocity, its personal-best skill, and the global-best skill to produce a new semantic velocity, which is then used to rewrite its current skill. (iv) The updated skills are evaluated on a validation batch, and the validation scores are used to update the personal-best and global-best skills. We visualize the overall pipeline of AgentPSO in Figure 1.

4.1 INDEPENDENT SOLVING AND PEER OBSERVATION

At iteration t, all agents first solve the same training batch independently using their current skills s_i^t *without* inter-agent communication. This independent solving phase allows us to evaluate the reasoning behavior induced by each skill before it is affected by peer interaction. In contrast, debate-based methods allow agents to exchange arguments during inference, which entangles their outputs and makes it harder to attribute the final result to a particular skill. In addition, AgentPSO maintains this non-interactive design at test time. The evolved agents solve problems in parallel and aggregate

Algorithm 1 AgentPSO

Require: Validation accuracy F, initial skills $\{s_i^0\}_{i=1}^N$, iterations T, margin ϵ
$\{v_i^0, p_i^0\} \leftarrow \{\text{“”}, s_i^0\}$ for all i
$g^0 \leftarrow s_{j^\star}^0$, where $j^\star = \arg\max_i F(s_i^0)$
for $t = 0$ to $T - 1$ **do**
  Each agent solves the training batch using s_i^t
  for $i = 1$ to N **do**
    Construct peer observation O_i^t
    $d_i^t \leftarrow \text{Reflect}(s_i^t, O_i^t)$
    $v_i^{t+1} \leftarrow \text{VelocityUpdate}(v_i^t, d_i^t, s_i^t, p_i^t, g^t)$
    $s_i^{t+1} \leftarrow \text{SkillUpdate}(s_i^t, v_i^{t+1})$
  end for
  Evaluate $\{s_i^{t+1}\}$ on a validation batch
  Update each p_i^{t+1} with s_i^{t+1} only if its validation score improves by more than ϵ
  Update g^{t+1} only when it improves global-best validation score by more than ϵ
end for
return $\{p_i^T\}_{i=1}^N$ and g^T

their answers by majority voting, avoiding the multi-round communication overhead of debate-based inference.

After all agents complete the batch, we construct a peer observation O_i^t for each agent. This observation contains the agent's own output and the peer agents' outputs, including final answers, reasoning traces, and correctness labels, while excluding the peer agents' skill instructions. This design encourages agents to learn from peer reasoning behaviors rather than copying peer skill texts. Consequently, each agent can identify what made another trajectory successful (*e.g.*, checking intermediate assumptions or decomposing the problem more carefully), and use these observations to improve its own skill.

4.2 PSO-Inspired Skill Evolution

Having constructed the peer observations, the next step is to update each agent's skill by integrating its own experience with the population's discoveries. To structure this integration, AgentPSO follows the update process based on the mechanics of Particle Swarm Optimization (PSO).

Standard PSO maintains, for each particle, its current position x_i^t, velocity v_i^t, personal-best position p_i^t, and swarm-level global-best position g^t. It updates each particle in two stages: first, it updates the velocity using the previous velocity, the particle's personal-best position, and the swarm's global-best position; then, it updates the particle's position by moving it according to the new velocity.

However, applying this update structure to natural-language skills requires an additional source of direction. In continuous PSO, the vector differences $p_i^t - x_i^t$ and $g^t - x_i^t$ provide explicit directions toward better positions. In natural-language skill space, however, knowing that a personal-best or global-best skill performed well does not directly explain which reasoning behavior should be added, removed, or strengthened. Without such behavioral guidance, the update tends to imitate the best skill state rather than learning a reusable reasoning principle.

To address this, AgentPSO introduces a self-reflective direction d_i^t derived from the peer observation O_i^t:

$$d_i^t = \text{Reflect}(s_i^t, O_i^t). \tag{1}$$

This direction tells the agent how to improve its skill, given how its own behavior compares with peers' on the same batch. Thus, d_i^t provides a concrete behavioral update direction without directly copying another agent's skill instruction.

AgentPSO retains the PSO two-stage update within the natural-language skill space. For agent A_i, the current skill s_i^t functions as the particle position, representing the candidate skill being optimized. The semantic velocity v_i^t acts as the particle velocity, representing the textual revision direction applied at the next step. The personal-best skill p_i^t and the global-best skill g^t correspond to the best skill previously found by A_i and the best skill discovered by the population, respectively.

The updated semantic velocity is generated as

$$v_i^{t+1} = \mathrm{VelocityUpdate}\big(v_i^t, d_i^t, s_i^t, p_i^t, g^t\big). \tag{2}$$

Here, $\mathrm{VelocityUpdate}$ replaces numerical vector differences with a semantic update operator. Given the previous velocity, the self-reflective direction, the current skill, and the personal-best and global-best skill references, the LLM synthesizes a new textual velocity that describes how the current skill should be revised.

Finally, AgentPSO performs the analogue of the PSO position update by rewriting the current skill according to the updated velocity:

$$s_i^{t+1} = \mathrm{SkillUpdate}\big(s_i^t, v_i^{t+1}\big). \tag{3}$$

Rather than adding a numerical velocity vector to a numerical position vector, $\mathrm{SkillUpdate}$ semantically rewrites the current skill. The updated skill builds on the agent's current skill rather than starting from scratch, and remains applicable to new problems. Detailed prompt templates are provided in Appendix F.

4.3 Validation-Based Best-State Selection

After generating the updated skills, AgentPSO evaluates each s_i^{t+1} on a validation batch using the validation accuracy F. If $F(s_i^{t+1})$ exceeds the score of the current personal-best skill p_i^t by more than ϵ, we set $p_i^{t+1} = s_i^{t+1}$; otherwise, we keep $p_i^{t+1} = p_i^t$. Likewise, the global-best skill is updated only if the strongest updated skill improves the current global-best score by more than ϵ. This margin ϵ prevents small or noisy fluctuations from immediately replacing previously effective skills.

At initialization, the semantic velocities are set to empty update directions, $v_i^0 =$ "". The personal-best skills are initialized as $p_i^0 = s_i^0$, and the initial global-best skill g^0 is selected as the highest-scoring initial skill on the validation batch. Algorithm 1 summarizes the overall procedure.

5 Experimental Setup

Datasets. We evaluate AgentPSO on five benchmarks covering mathematical and general reasoning. (1) For mathematical reasoning, we use DeepMath He et al. (2025), MATH Lightman et al. (2023), AIME25 Zhang & Math-AI (2025), and Minerva Lewkowycz et al. (2022), which require multi-step reasoning, symbolic manipulation, and numerical problem solving. The resulting optimized agent skills are evaluated on the DeepMath test set and further applied to out-of-distribution mathematical benchmarks, including MATH, AIME25, and Minerva. (2) For general reasoning, we use BigBenchHard (BBH) Suzgun et al. (2023), a challenging benchmark composed of 23 diverse reasoning tasks, including logical, symbolic, commonsense, and multi-step reasoning. We randomly split BBH into training, validation, and test sets and report the final performance on the BBH test split. Further details on the datasets and splits are provided in Appendix B.

Baselines. We compare AgentPSO with baselines from four main categories. The implementation details of all baselines are described in Appendix C:

- **Vanilla single-agent methods.** We include Chain-of-Thought Prompting (CoT) Wei et al. (2022) and Step-Back Prompting Zheng et al. (2024) as basic single-agent reasoning baselines.
- **Advanced single-agent methods.** We include Self-Refine Madaan et al. (2023), Reflection Shinn et al. (2023), Self-Consistency Wang et al. (2023), and Tree-of-Thoughts Yao et al. (2023), which improve single-agent reasoning through refinement, reflection, sampling, or structured search.
- **Multi-agent methods.** We compare against MAD Du et al. (2023), MoA Wang et al. (2025a), and DMAD Liu et al. (2025a), which leverage multiple agents to improve reasoning through debate, aggregation, or diversity.
- **Self-evolving methods.** We further include DSPy Khattab et al. (2024) and GEPA Agrawal et al. (2026) as prompt optimization baselines, and ADAS Hu et al. (2024) as an automated agentic-system design baseline.

Table 2: **Comparison with single-agent, multi-agent, and self-evolving baselines on GPT-5.4-mini.** All reported values denote mean accuracy ± standard deviation over three random seeds. AgentPSO achieves the best average performance across all benchmarks.

Category	Method	GPT-5.4-mini					
		DeepMath	MATH	AIME25	Minerva	BBH	Avg.
Vanilla Single-Agent	Chain-of-Thought	68.83 ± 0.62	90.20 ± 0.16	38.89 ± 1.57	38.97 ± 1.08	85.33 ± 1.03	64.44
	Step-Back Prompting	65.83 ± 3.40	89.87 ± 0.75	47.78 ± 3.14	37.62 ± 1.65	84.17 ± 1.70	65.05
Advanced Single-Agent	Self-Refine	67.67 ± 2.25	90.40 ± 0.86	51.11 ± 1.57	36.27 ± 1.51	83.83 ± 0.94	65.86
	Reflection	68.67 ± 2.72	90.60 ± 0.91	44.44 ± 4.16	38.73 ± 2.13	83.17 ± 0.62	65.12
	Self-Consistency	68.33 ± 1.04	89.13 ± 0.42	54.44 ± 5.09	28.19 ± 1.12	**86.33** ± 0.76	65.28
	ToT	63.17 ± 1.76	81.60 ± 0.80	52.22 ± 3.85	30.15 ± 2.30	78.83 ± 1.04	61.19
Multi-Agent	MAD	70.00 ± 1.00	**95.93** ± 0.12	**56.67** ± 0.00	33.82 ± 1.10	82.33 ± 1.44	67.75
	MoA	71.00 ± 1.00	95.53 ± 0.12	55.56 ± 1.92	28.68 ± 1.91	78.67 ± 2.47	65.89
	DMAD	71.50 ± 1.32	95.67 ± 0.61	55.56 ± 1.92	35.17 ± 0.56	83.67 ± 1.89	68.31
Self-Evolving	DSPy	74.67 ± 3.21	92.27 ± 0.42	52.22 ± 5.09	36.52 ± 0.93	83.50 ± 5.07	67.84
	GEPA	77.83 ± 1.76	91.67 ± 1.03	48.89 ± 5.09	39.95 ± 3.08	82.17 ± 4.04	68.10
	ADAS	62.33 ± 2.31	79.67 ± 2.01	54.44 ± 1.92	30.76 ± 2.22	84.17 ± 1.04	62.27
AgentPSO	AgentPSO	**79.00** ± 0.87	94.00 ± 0.60	**56.67** ± 0.00	**41.79** ± 0.93	**86.33** ± 0.58	**71.56**

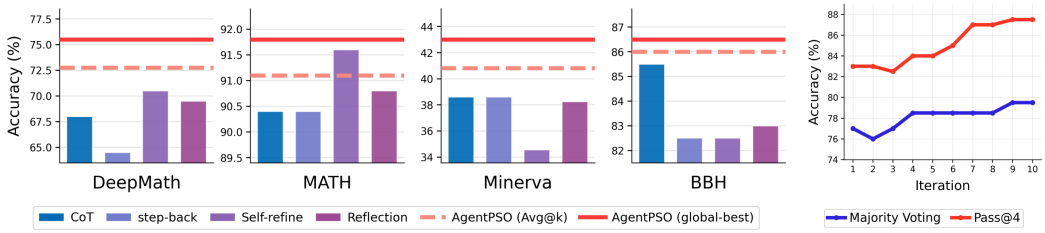


Figure 2: **Progressive improvement of AgentPSO-evolved skills.** (Left) Evolved personal-best and global-best skills outperform the initial skills on average. (Right) DeepMath performance improves over iterations, showing gradual refinement of the agent population.

Implementation Details. We implement AgentPSO with ChatGPT (gpt-5.4-mini) as the backbone model. All experiments use four agents initialized with different CoT-based reasoning methods. We optimize agent skill states for 100 training samples over 10 iterations, without updating backbone model parameters. Detailed hyperparameter settings and prompts are provided in Appendices A.2 and F.

6 Main Results

6.1 AgentPSO Improves Multi-Agent Reasoning

Main performance comparison. To evaluate the effectiveness of AgentPSO's skill evolution, we compare it against vanilla single-agent, advanced single-agent, multi-agent, and self-evolving baselines across GPT-5.4-mini with three random seeds (Table 2). AgentPSO achieves the best average performance across the five benchmarks (71.56%), outperforming single-agent prompting methods, static multi-agent baselines, and self-evolving methods. This pattern suggests that AgentPSO does not merely improve the in-distribution optimization benchmark but also learns skills that generalize to different mathematical and general reasoning benchmarks.

Progressive improvement of evolved agent skills. To examine whether AgentPSO genuinely improves agent skills through iterative evolution, we analyze the personal-best and global-best skills over the optimization process. Figure 2 (Left) compares the initial agent skills with the learned personal-best and global-best skills. After optimization, both the average personal-best agents and the global-best agent outperform the initial agents on average. We find that the global-best agent achieves the strongest single-agent performance, indicating AgentPSO drives the agent population toward a more effective skill configuration. Figure 2 (Right) shows how performance changes across optimization iterations. AgentPSO improves the population gradually rather than converging to a local optimum. As the iterations progress, the collective performance of the saved personal-best agents steadily improves, with the final iterations achieving the strongest results.

Table 3: **Cross-benchmark transfer of learned reasoning skills.**

Dataset	Method	Accuracy	Pass@k	Avg@k
DeepMath	MAD	69.00	70.50	68.00
	AgentPSO$_{\text{DeepMath}}$	79.50	87.00	72.75
	AgentPSO$_{\text{BBH}}$	75.00	81.00	66.13
BBH	MAD	84.00	87.00	81.88
	AgentPSO$_{\text{BBH}}$	87.00	93.00	85.88
	AgentPSO$_{\text{DeepMath}}$	85.50	90.50	82.75

Table 4: **Cross-model transfer of AgentPSO-learned skills from GPT to Claude.**

Method	DeepMath	MATH	AIME25	Minerva	BBH	Avg.
Chain-of-Thought	60.00	90.60	30.00	36.03	47.50	52.83
Step-Back Prompting	63.50	89.80	36.67	**36.40**	51.00	55.47
Self-Refine	61.00	**91.60**	30.00	34.56	50.00	53.43
Reflection	60.00	90.20	36.67	35.66	55.00	55.51
AgentPSO (GPT→Claude)	**70.50**	91.40	**40.00**	33.46	**56.50**	**58.37**

6.2 AGENTPSO LEARNS TRANSFERABLE REASONING SKILLS

Cross-domain transfer. To test whether AgentPSO discovers generalizable reasoning rather than overfitting to a particular dataset, Table 3 evaluates cross-benchmark transfer of AgentPSO-learned skills, where Pass@k denotes whether at least one of the k agents produces the correct answer and Avg@k denotes the average accuracy across the k individual agents. We denote AgentPSO skills optimized on BigBenchHard and DeepMath as AgentPSO$_{\text{BBH}}$ and AgentPSO$_{\text{DeepMath}}$, respectively. Although skills optimized on the target benchmark achieve the strongest performance, cross-trained AgentPSO skills still outperform the static MAD baseline on both DeepMath and BBH. This trend is observed not only in majority-vote accuracy but also in pass@k, suggesting that the evolved agent population retains useful reasoning behaviors even when transferred to a different benchmark. These results suggest that AgentPSO learns reasoning procedures that transfer across reasoning domains, not just skills tied to the training data.

Cross-model transfer. Table 4 evaluates whether AgentPSO's skills transfer across different backbone LLMs. We obtain the final skill population optimized through AgentPSO using GPT, then we directly apply these skills to Claude (claude-haiku-4-5-20251001) at test time without any additional training or adaptation. The single-agent baselines report the performance of Claude using the initial skill prompts, including Chain-of-Thought, Step-Back Prompting, Self-Refine, and Reflection. AgentPSO (GPT→Claude) reports the performance of Claude when using the GPT-optimized AgentPSO skill. Overall, the GPT-optimized AgentPSO skill improves Claude's average accuracy to 58.37%, outperforming all initial skill baselines. These results provide preliminary evidence that AgentPSO-learned skills can improve another backbone LLM. Additional GPT-to-Qwen transfer and cross-framework transfer results are provided in Appendices D.5 and D.6.

6.3 ANALYSIS OF SKILL EVOLUTION

Table 5: **Comparison of skill-evolution stability with and without velocity.**

Method	Acc.(%)	Mean	Median	Max	High(%)
w/o velocity	78.0	0.0965	0.0562	0.4486	32.5
w/ velocity	**79.5**	**0.0868**	**0.0488**	**0.4153**	**15.0**

Effect of velocity on skill stability. We further analyze the role of the velocity term in AgentPSO. In the no-velocity variant, we remove the velocity update step in Eq. (2) and directly update the skill using Eq. (3). Given the skill s_t and the updated skill s_{t+1}, we compute $\text{Drift}(s_t, s_{t+1}) = 1 - \cos\left(\text{Emb}(s_t), \text{Emb}(s_{t+1})\right)$, where $\text{Emb}(\cdot)$ is computed using OpenAI embedding API[1]. A larger drift indicates a larger semantic change in the updated skill. As shown in Table 5, removing velocity decreases the final accuracy from 79.50% to 78.00%. Moreover, velocity makes skill evolution smoother, reducing the mean semantic drift and the proportion of high-drift updates, defined as drift > 0.1. These results suggest that velocity improves performance and stabilizes evolution by reducing abrupt semantic shifts.

Qualitative analysis of skill evolution. We qualitatively examine how AgentPSO updates the velocities and skills of individual agents. Figures 6 and 7 in Appendix E.2 show the evolution of the agent initialized with a Self-Refine skill. In the early stage, the updates mainly improve answer formatting, such as checking the required output type, using canonical notation, and verifying edge cases. These changes are primarily driven by the self-reflective direction, which helps the agent correct recurring formatting and validation errors. In the middle stage, the agent incorporates guidance from the global-best skill, adding higher-level reasoning behaviors such as problem-

[1] https://developers.openai.com/api/docs/models/text-embedding-3-small

type classification and method selection. By the later stage, the skill becomes a hybrid policy that preserves the original Self-Refine structure while integrating global-best abstraction and feedback-derived validation behavior. Overall, these analyses show that AgentPSO progressively transforms simple initial instructions into structured procedural policies that encode concrete behaviors such as answer-format control, verification, and strategy selection.

6.4 Ablation Study

Table 6: **AgentPSO component ablation on DeepMath.**

Input Components	Acc.	Pass@k
$v_i \oplus s_i \oplus d_i \oplus p_i \oplus g$	79.50	87.00
$v_i \oplus s_i \oplus p_i \oplus g$	72.50	80.50
$v_i \oplus s_i \oplus p_i$	73.00	80.00
$v_i \oplus s_i \oplus g$	72.00	81.50
$v_i \oplus s_i \oplus d_i$	76.00	81.50

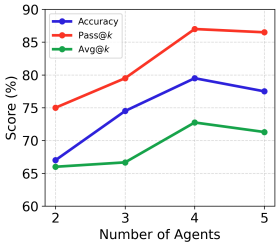


Figure 3: **Effect of the number of agents.**

Contribution of self-reflective direction. Table 6 highlights the importance of the self-reflective direction (d_i) in AgentPSO. When d_i is removed from the full update rule, performance drops substantially, indicating that standard PSO-style guidance alone is insufficient for fully optimizing agent skills. Personal-best guidance (p_i) and global-best guidance (g) provide useful signals by preserving and propagating successful skills, but they are most effective when combined with d_i. In contrast, the variant that combines the velocity with the self-reflective direction achieves stronger performance than the variants based only on personal-best or global-best guidance. This suggests that feedback derived from peer reasoning trajectories provides a more direct and informative signal for improving agent skills.

Furthermore, we analyze the same update-component settings from a functional perspective in Appendix E.3. The analysis shows that full AgentPSO produces more frequent and functionally diverse skill updates, while removing d_i leads to substantially more limited evolution. Additional validation trajectories comparing AgentPSO with and without d_i are provided in Appendix D.1.

Effect of the number of agents. Figure 3 analyzes how the number of agents affects AgentPSO performance on DeepMath. Increasing the number of agents from $k = 2$ to $k = 4$ consistently improves performance. This shows that agent diversity is beneficial, as a larger population provides more reasoning trajectories and candidate skills for evolution. However, performance slightly drops when using $k = 5$, suggesting that additional agents can introduce redundancy and make the collective update process harder to stabilize. Therefore, we use $k = 4$ as the default number of agents, which provides the best trade-off between diversity and stability.

7 Conclusion

We proposed **AgentPSO**, a PSO-inspired framework that evolves externalized reasoning skills represented as natural-language instructions. Beyond the standard PSO components of personal-best and global-best guidance, AgentPSO introduces a self-reflective direction that allows agents to extract reusable lessons from peer reasoning trajectories. By combining these update signals, AgentPSO refines each agent's skill without updating the backbone model parameters. The learned skills exhibit strong transferability, generalizing across both reasoning benchmarks and backbone LLMs. This suggests that AgentPSO learns reusable reasoning behaviors instead of task- or model-specific prompting patterns. Overall, our results show that multi-agent collaboration can serve not only as an inference-time mechanism for answer refinement, but also as a model-parameter-free process for evolving reusable reasoning skills at the population level. This may improve the reliability and efficiency of reasoning systems by enabling agents to acquire reusable reasoning behaviors without parameter updates.

Limitations. The current experiments focus on reasoning benchmarks. Although AgentPSO avoids parameter updates, it requires additional training-time LLM calls to evolve the agent population. Another limitation is that evolved agent skills may amplify incorrect reasoning patterns if the feedback, peer observations, or validation signals are flawed. Moreover, AgentPSO optimizes reasoning skills rather than acquiring new domain knowledge. Future work may combine AgentPSO with stronger verification or retrieval mechanisms, especially for knowledge-intensive domains.

AI USE STATEMENT

Generative AI models were used as core experimental components of this work. In particular, large language models were used to instantiate the agents in AgentPSO, generate reasoning trajectories, derive self-reflective directions, produce semantic velocity updates, and revise agent skill instructions. Additional language models were used in the cross-model transfer experiments described in the paper. These uses constitute part of the method being studied and are documented in the main text and appendices.

Generative AI tools were also used during the research process to assist with refinement of experimental designs, implementation and debugging, and editing the manuscript for clarity and readability. Generative AI was not used to fabricate experimental data, results, or citations. All methodological decisions, experimental configurations, analyses, and scientific claims were made or independently verified by the authors.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Mohammad Almansoori, Komal Kumar, and Hisham Cholakkal. Self-evolving multi-agent simulations for realistic clinical interactions. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2025.

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F. Survey Certification.

Anthropic. Models overview. https://platform.claude.com/docs/en/about-claude/models/overview, 2026. Accessed: 2026-05-06.

Nafew Azim, Abrar Ur Alam, Hasan Bin Omar, Abdullah Mohammad Muntasir Adnan Jami, Jawad Ibn Ahad, Muhammad Rafsan Kabir, Md. Ismail Hossain, Fuad Rahman, Mohammad Ruhul Amin, Shafin Rahman, and Nabeel Mohammed. AutoDSPy: Automating modular prompt design with reinforcement learning for small and large language models. In Saloni Potdar, Lina Rojas-Barahona, and Sebastien Montella (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2881–2896, Suzhou (China), November 2025. Association for Computational Linguistics. ISBN 979-8-89176-333-3. doi: 10.18653/v1/2025.emnlp-industry.192. URL https://aclanthology.org/2025.emnlp-industry.192/.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. Benchmarking large language models on answering and explaining challenging medical questions, 2024. URL https://arxiv.org/abs/2402.18060.

Yu Cui, Hang Fu, Haibin Zhang, Licheng Wang, and Cong Zuo. Free-MAD: Consensus-free multi-agent debate, 2026. URL https://openreview.net/forum?id=46jbtZZWen.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Zhaopeng Feng, Jiayuan Su, Jiamei Zheng, Jiahan Ren, Yan Zhang, Jian Wu, Hongwei Wang, and Zuozhu Liu. M-MAD: Multidimensional multi-agent debate for advanced machine translation evaluation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7084–7107, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.351. URL https://aclanthology.org/2025.acl-long.351/.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2024. URL https://openreview.net/forum?id=HKkiX32Zw1.

Daniel Freitas, Luis G. Lopes, and Fernando Morgado-Dias. Particle swarm optimisation: A historical review up to the current developments. *Entropy*, 22(3):362, 2020.

Lingyue Fu, Xin Ding, Yaoming Zhu, Shao Zhang, Lin Qiu, Weiwen Liu, Weinan Zhang, Xuezhi Cao, Xunliang Cai, Jiaxin Ding, and Yong Yu. CATArena: Evaluation of LLM agents through iterative tournament competitions, 2026. URL https://openreview.net/forum?id=aSr8eoe2vz.

Ahmed G. Gad. Particle swarm optimization algorithm and its applications: A systematic review. *Archives of Computational Methods in Engineering*, 29:2531–2561, 2022. doi: 10.1007/s11831-021-09694-4.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. 2025. URL https://arxiv.org/abs/2504.11456.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Voting or consensus? decision-making in multi-agent debate. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11640–11671, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.606. URL https://aclanthology.org/2025.findings-acl.606/.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pp. 1942–1948 vol.4, 1995. doi: 10.1109/ICNN.1995.488968.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations (ICLR)*, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Zijie Lin and Bryan Hooi. Enhancing multi-agent debate system performance via confidence expression. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 6453–6471, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.343. URL https://aclanthology.org/2025.findings-emnlp.343/.

Shuai Ling, Lizi Liao, Dongmei Jiang, and Weili Guan. MeMAD: Structured memory of debates for enhanced multi-agent reasoning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=zLbmsdyTiN.

Tongxuan Liu, Xingyu Wang, Weizhe Huang, Wenjiang Xu, Yuting Zeng, Lei Jiang, Hailong Yang, and Jing Li. Groupdebate: Enhancing the efficiency of multi-agent debate using group discussion. *arXiv preprint arXiv:2409.14051*, 2024. doi: 10.48550/arXiv.2409.14051.

Yexiang Liu, Jie Cao, Zekun Li, Ran He, and Tieniu Tan. Breaking mental set to improve reasoning through diverse multi-agent debate. In *ICLR*, 2025a.

Yuanzhe Liu, Ryan Deng, Tim Kaler, Xuhao Chen, Charles E Leiserson, Yao Ma, and Jie Chen. Lessons learned: A multi-agent framework for code LLMs to learn and improve. In *Advances in Neural Information Processing Systems 38*, 2025b.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

OpenAI. Introducing gpt-5.5. https://openai.com/ko-KR/index/introducing-gpt-5-5/, 2026. Accessed: 2026-05-06.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. 2022.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Yamato Shinohara, Jinglue Xu, Tianshui Li, and Hitoshi Iba. Large language models as particle swarm optimizers. In *2025 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–4, 2025. doi: 10.1109/CEC65147.2025.11043021.

Jinwei Su, Yinghui Xia, Yiqun Duan, Jun Du, Jianuo Huang, Tianyu Shi, and Lewei He. Debflow: Automating agent creation via agent debate. *arXiv preprint arXiv:2503.23781*, 2025. doi: 10.48550/arXiv.2503.23781.

James Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies, and Nations*. Doubleday, New York, 2004. URL https://psycnet.apa.org/record/2004-20179-000.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL (Findings)*, pp. 13003–13051, 2023. URL https://doi.org/10.18653/v1/2023.findings-acl.824.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=h0ZfDIrj7T.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=NTAhi2JEEE.

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

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Zhiyuan Weng, Guikun Chen, and Wenguan Wang. Do as we do, not as you think: the conformity of large language models. *arXiv preprint arXiv:2501.13381*, 2025.

Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. Comas: Co-evolving multi-agent systems via interaction rewards. *arXiv preprint arXiv:2510.08529*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pUpzQZTvGY.

Kai Zhang, Qiujun Huang, and Yimin Zhang. Enhancing comprehensive learning particle swarm optimization with local optima topology. *Information Sciences*, 471:1–18, 2019. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2018.08.049. URL https://www.sciencedirect.com/science/article/pii/S0020025518306674.

Yao Zhang, Chenyang Lin, Shijie Tang, Haokun Chen, Shijie Zhou, Yunpu Ma, and Volker Tresp. Swarmagentic: Towards fully automated agentic system generation via swarm intelligence. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1778–1818, 2025.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3bq3jsvcQ1.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=WZH7099tgfM.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. Self-discover: Large language models self-compose reasoning structures. *arXiv preprint arXiv:2402.03620*, 2024. doi: 10.48550/arXiv.2402.03620.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=92gvk82DE-.

Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. Medxpertqa: Benchmarking expert-level medical reasoning and understanding. *arXiv preprint arXiv:2501.18362*, 2025.

A Experimental Details

Table 7: **Hyperparameter settings for AgentPSO.**

Hyperparameter	Value
Number of agents	4
Number of iterations	10
Training/Validation pool size	100 / 100
Training/Validation batch size	10 / 20
Epsilon margin	0.01
Maximum velocity length	200 words / 300 tokens
Maximum skill length	1200 words / 800 tokens

Table 8: **Dataset statistics used in our experiments.**

Dataset	Train	Validation	Test
DeepMath He et al. (2025)	100	100	200
BigBenchHard Suzgun et al. (2023)	100	100	200
MATH Lightman et al. (2023)	–	–	500
AIME25 Zhang & Math-AI (2025)	–	–	30
Minerva Lewkowycz et al. (2022)	–	–	272

A.1 Instruction of Agents

We initialize the agent population with four distinct reasoning skill modules. Each agent is assigned one initial instruction that encourages a different reasoning behavior. These initial skills are used to induce diversity among agents before the optimization process begins. The full instructions are as follows.

- **Agent 1: Chain of Thought.** *Solve the problem step by step.*
- **Agent 2: Step-Back Prompting.** *Before solving, step back and identify the general principle or problem type.*
- **Agent 3: Self-Refine.** *First solve the problem, then review and improve the solution.*
- **Agent 4: Reflection.** *Solve while reflecting on assumptions and possible failure points.*

A.2 Details of Hyperparameters

Since AgentPSO operates in natural-language skill space instead of a numeric vector space, we do not sample stochastic PSO coefficients. Instead, all natural-language update components are incorporated with equal weight. The primary hyperparameters of AgentPSO control the number of agents, the sizes of the training and validation pools, the batch sizes used during optimization, the epsilon margin for updating best skills, and the maximum lengths of generated velocity and skill descriptions. Table 7 summarizes the hyperparameter settings.

B Datasets

Table 8 summarizes the dataset splits used in our experiments. For DeepMath and BigBenchHard (BBH), we use 100 training samples, 100 validation samples, and 200 test samples. The training split is used for skill updates, the validation split is used for personal-best and global-best selection, and the test split is used only for final evaluation. To reduce API cost and computational overhead, we randomly sample 200 test examples, following prior multi-agent reasoning studies Liu et al. (2025a); Du et al. (2023). For MATH, AIME25, and Minerva, we perform transfer evaluation without additional training or validation.

Validation scheduler. For DeepMath and BBH, the 100 validation samples are divided into five non-overlapping subsets of 20 samples. At each iteration, validation is conducted on one subset. If an agent achieves a perfect score on the current validation subset, the next iteration uses a different subset. This scheduler reduces overfitting to a single saturated validation subset and encourages robust best-skill selection.

C Baseline Details

Vanilla single-agent methods

- Chain-of-Thought Prompting (CoT) Wei et al. (2022): It is instructed to solve the problem step by step, decomposing the problem into intermediate reasoning steps before producing the final answer. We use the same prompt as Agent 1 in Appendix A.1.
- Step-Back Prompting Zheng et al. (2024): It is encouraged to first abstract away from the specific problem and identify the underlying principle, concept, or problem type before solving. We use the same prompt as Agent 2 in Appendix A.1.

Advanced single-agent methods

- Self-Refine Madaan et al. (2023): It first generates an initial solution and then reviews and improves its own reasoning before finalizing the answer. We use the same prompt as Agent 3 in Appendix A.1.
- Reflection Shinn et al. (2023): It solves the problem while reflecting on assumptions, uncertainty, and possible failure points. We use the same prompt as Agent 4 in Appendix A.1.
- Self-Consistency Wang et al. (2023): For each problem, we prompt the model with *"Let's think step by step"* and generate 4 independent reasoning paths using stochastic decoding. We extract the final answer from each generated path and determine the final prediction by majority voting over the 4 extracted answers.
- Tree-of-Thoughts Yao et al. (2023): We implement a Tree-of-Thoughts baseline using breadth-first search. For each problem, the model expands the current reasoning states by generating 4 candidate thoughts. Each partial solution is then evaluated by the same model, which serves as a value function for estimating the promise of the current reasoning state. We prune the search tree to a beam width of 2 at each depth and repeat this process up to a maximum depth of 3. After the search reaches the maximum depth, the remaining states are completed into final solutions. The final prediction is selected as the answer from the highest-scoring completed solution.

Multi-agent methods

- MAD Du et al. (2023): We implement a 4-agent, 2-round Multi-Agent Debate baseline. In the first round, each agent independently solves the problem. In the second round, each agent observes the other agents' first-round responses and produces a revised final answer. We use majority voting over the second-round answers as the final prediction.
- MoA Wang et al. (2025a): We implement a 3-layer Mixture-of-Agents baseline. In Layer 1, 4 proposer agents independently solve the problem. In Layer 2, 4 aggregator agents each receive the original problem and all Layer-1 outputs, and synthesize improved solutions. In Layer 3, a single final aggregator receives the original problem and all Layer-2 outputs, and produces the final answer. We use the final aggregator's output as the final prediction.
- DMAD Liu et al. (2025a): We implement a 4-agent, 2-round Diverse Multi-Agent Debate baseline. Each agent is assigned a distinct reasoning method: Chain-of-Thought, Step-Back Prompting, Self-Refine, or Reflection. In the first round, each agent independently solves the problem using its assigned method. In the second round, each agent observes the other agents' first-round solutions and produces a refined answer while maintaining its assigned method. We use majority voting over the second-round answers as the final prediction.

Self-evolving methods

- DSPy Khattab et al. (2024): We implement a single-agent Chain-of-Thought baseline using DSPy[2] and optimize its prompt and configuration with MIPROv2. DSPy represents an LM pipeline as a declarative program with input–output signatures and compiles it by optimizing the prompts and demonstrations used by its modules. We use MIPROv2 to optimize this single-agent reasoning program on the training set, and use the optimized program directly at test time.

[2] https://dspy.ai/

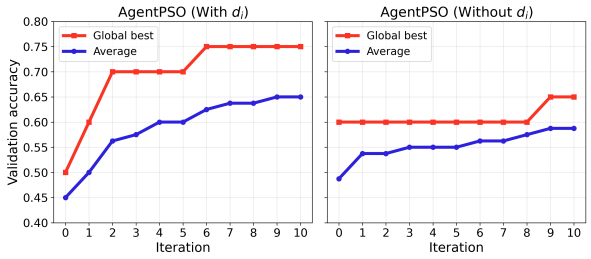


Figure 4: **Validation accuracy of the global-best and average accuracy over iterations.** The global-best trajectory improves steadily, while the variant without d_i shows only limited progress.

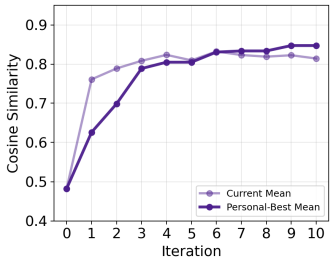


Figure 5: **Semantic alignment with the final global-best skill.** Agent skills become increasingly similar to the final global-best skill over iterations.

- GEPA Agrawal et al. (2026): We implement a single-agent prompt optimization baseline using the GEPA API[3]. GEPA is a reflective prompt optimizer that improves textual components of an LM system using execution traces, feedback, and LLM-based reflection.
- ADAS Hu et al. (2024): ADAS uses a meta-agent to iteratively generate new executable agent designs from an archive of previously evaluated agents. We initialize the archive with several hand-designed agents, including Chain-of-Thought, Self-Consistency, Self-Refine, LLM Debate, Quality-Diversity, Step-back Abstraction, and Role Assignment. Each candidate agent is evaluated on the validation/search set and added to the archive. After search, we select the best validation agent and evaluate it on the held-out test set.

D Additional Analysis

D.1 Effect of the self-reflective direction

To isolate the contribution of the self-reflective direction d_i, Figure 4 compares the full AgentPSO update (With d_i) and a PSO-style variant (Without d_i). While both variants benefit from PSO-style guidance, AgentPSO with d_i achieves better performance than the variant without d_i throughout optimization. In contrast, removing d_i leads to weaker progress, suggesting that personal-best and global-best guidance alone are not sufficient to fully drive skill evolution. These results indicate that the self-reflective direction d_i helps each agent learn from its own mistakes and from peer agents' reasoning, leading to better skill updates over iterations.

D.2 Skill alignment dynamics

To examine how the agent population evolves relative to the final best skill in the DeepMath experiment, we measure the mean cosine similarity between each agent's skill and the final global-best skill across iterations. Embeddings are computed using the OpenAI embedding API[4] and averaged over agents using DeepMath benchmark. As shown in Figure 5, the similarity rises in the early iterations and then plateaus well below 1, indicating that agents partially converge toward the global-best skill but do not collapse onto it. This behavior is consistent with the intended role of AgentPSO: the population-level guidance steers agents toward effective regions of the skill space, while the self-reflective direction and per-agent personal-best preserve diversity across the swarm.

D.3 Optimization and Inference Cost

We compare the optimization and inference costs of AgentPSO with single-agent, multi-agent, and self-evolving baselines. Table 9 details the total optimization time over the full training set, alongside the average inference latency measured across 30 DeepMath test cases.

[3] https://gepa-ai.github.io/gepa/

[4] https://developers.openai.com/api/docs/models/text-embedding-3-small

Table 9: **Optimization and inference cost comparison.** Optimization time is measured over the full optimization set, and inference latency is averaged over 30 DeepMath test examples.

Method	Optimization Time (min)	Inference Time (sec/case)
Single Agent	–	5.40
MAD	–	42.72
MoA	–	52.45
DSPy	26.4	5.35
GEPA	19.2	4.52
ADAS	1469.1	18.26
AgentPSO	**190.6**	**22.95**

Table 10: **Comparison under matched LLM-call budgets on DeepMath.** The budget includes both optimization and test-time inference calls.

Method	# LLM Calls	Acc. (%)
Self-Consistency	2,200	72.50
MAD	2,400	69.50
GEPA	≈2,240	77.50
ADAS	≈4,700	61.00
AgentPSO	**2,200**	**79.50**

Training-free methods such as the single-agent baseline, MAD, and MoA require no separate optimization stage. However, MAD and MoA incur substantially higher inference latency, requiring 42.72 and 52.45 seconds per case, respectively, compared with 5.40 seconds for the single-agent baseline. In contrast, while AgentPSO introduces an offline skill-optimization stage taking 190.6 minutes, it significantly reduces inference latency to 22.95 seconds per case. It achieves this efficiency by deploying optimized agents independently at test time, bypassing computationally expensive multi-round debates or layer-wise aggregation.

Among self-evolving methods, DSPy and GEPA require substantially less optimization time and achieve lower inference latency because they use a single optimized program or prompt at test time, but their final average performance is lower than AgentPSO. ADAS maintains a competitive inference latency of 18.26 seconds per case but suffers from a prohibitively high optimization cost of 1469.1 minutes. Overall, AgentPSO introduces additional offline optimization cost in exchange for lower test-time latency than debate-based multi-agent methods, while retaining multi-agent reasoning through the optimized agent population.

D.4 Comparison under Matched LLM-Call Budgets

To determine whether AgentPSO's performance gains simply stem from a higher number of LLM calls during skill optimization, we evaluate the methods under matched or closely comparable total LLM-call budgets. We count all LLM calls used across initialization, optimization, and test-time inference. For AgentPSO on DeepMath dataset, the total budget is 2,200 calls. Specifically, the initial validation requires 80 calls (4 agents $\times$ 20 validation examples); the iterative optimization requires 1,320 calls (10 iterations $\times$ 4 agents $\times$ [10 training examples $+$ 3 update operations $+$ 20 validation examples]); and the final test-time inference requires 800 calls (4 agents $\times$ 200 test examples).

To ensure a fair comparison, we scaled up the computational budgets of the baseline methods; Self-Consistency to use 11 parallel samples per test instance, extended MAD's inference to three debate rounds, and allocated a larger optimization budget to GEPA. As shown in Table 10, AgentPSO achieves the highest accuracy of 79.50% under these matched or closely comparable budgets. Notably, ADAS consumes substantially more calls yet yields considerably lower accuracy. These findings confirm that AgentPSO's superiority cannot be explained solely by the additional LLM calls used for optimization. Instead, the results demonstrate that population-guided skill evolution makes far more efficient use of the available computational budget.

D.5 Cross-model transfer: GPT to Qwen3-32B

To test whether GPT-discovered skills can transfer to an open-weight backbone, we directly apply the best skill optimized with GPT to Qwen3-32B without additional training or adaptation. Table 11 compares this transferred skill with Qwen3-32B single-agent baselines using the initial skill prompts. The single-agent baselines use Qwen3-32B with the initial skill prompts, while AgentPSO (GPT→Qwen) uses the GPT-optimized AgentPSO skill. The transferred skill improves

Table 11: **Cross-model transfer of AgentPSO-learned skills from GPT to Qwen3-32B.**

Method	DeepMath	MATH	AIME25	Minerva	BBH	Avg.
Chain-of-Thought	60.00	77.20	10.00	37.50	71.50	51.24
Step-Back Prompting	51.50	63.80	6.67	37.87	66.50	45.27
Self-Refine	62.00	78.40	6.67	**41.54**	71.00	51.92
Reflection	57.00	74.00	13.33	41.18	74.00	51.90
AgentPSO (GPT→Qwen)	**80.00**	**86.60**	**20.00**	37.87	**75.00**	**59.89**

Table 12: **Reusing AgentPSO-trained skills in multi-agent inference methods.** AgentPSO-trained skills are plugged into MAD and MoA at test time.

Dataset	MAD	AgentPSO + MAD	MoA	AgentPSO + MoA
DeepMath	69.00	76.50 (+7.50)	70.00	72.50 (+2.50)
Minerva	32.72	36.40 (+3.68)	27.57	29.78 (+2.21)
BigBenchHard	84.00	84.00 (+0.00)	77.50	82.50 (+5.00)

average accuracy to 59.89%, outperforming all initial skill baselines. These results suggest that AgentPSO-learned skills capture transferable reasoning behaviors that can improve another backbone LLM without additional optimization.

D.6 Cross-framework transfer: AgentPSO to multi-agent inference methods

To examine whether AgentPSO-learned skills can serve as reusable skill modules for existing multi-agent frameworks, we plug the optimized skills into MAD and MoA at test time. Table 12 shows that replacing the original prompts with AgentPSO-trained skills improves performance on most benchmarks. For MAD, all agents are instantiated with the global-best skill discovered by AgentPSO, while for MoA, we use different personal-best skills to preserve diversity among proposer agents. Across both MAD and MoA, replacing the original prompts with AgentPSO-trained skills improves performance on most benchmarks. This suggests that AgentPSO learns reusable agent skills that are not limited to a single inference procedure, but can be integrated into different multi-agent reasoning frameworks.

D.7 Effect of skill evolution.

To isolate the contribution of skill evolution from multi-agent diversity and majority voting, we compare AgentPSO with a static four-agent ensemble that uses the same initial reasoning skills and the same majority-voting inference procedure, but without skill evolution. As shown in Table 13, AgentPSO consistently outperforms this baseline across all five benchmarks, improving the average accuracy from 66.35% to 71.98%, demonstrating the benefit of evolving agent reasoning skills beyond static multi-agent ensembling.

D.8 Additional medical benchmark

To further examine whether AgentPSO can be applied beyond mathematical and general reasoning tasks, we conduct additional experiments on medical question-answering benchmarks. We use MedXpertQA Zuo et al. (2025) and MedBullets Chen et al. (2024), which contain challenging medical exam-style questions requiring domain-specific reasoning. AgentPSO is optimized on MedXpertQA using the same four-agent setting as in the main experiments, with 100 training examples and 200 test examples. We then evaluate the optimized agents on both MedXpertQA and MedBullets.

Table 14 summarizes the results. On MedXpertQA and MedBullets, AgentPSO improves pass@k over MAD, indicating that the optimized agent population is more likely to contain at least one correct solution among its agents. However, its majority-voting accuracy is lower than MAD.

Unlike mathematical or general reasoning benchmarks, medical QA is often constrained more by the model's domain-specific knowledge reliability than by reasoning diversity alone. Since AgentPSO

Table 13: **Effect of skill evolution:** comparison with static multi-agent majority voting.

Method	DeepMath	MATH	AIME25	Minerva	BBH	Avg.
Multi-Agent Majority Voting	72.50	93.00	46.67	34.56	85.00	66.35
AgentPSO	**79.50**	**94.60**	**56.67**	**42.65**	**86.50**	**71.98**

Table 14: **AgentPSO on medical question-answering benchmarks.** AgentPSO is optimized on MedXpertQA and evaluated on MedXpertQA and MedBullets. Results show strong pass@k performance, indicating improved coverage of correct solutions among agents.

Method	MedXpert		MedBullets	
	Accuracy	Pass@k	Accuracy	Pass@k
Chain-of-Thought	20.50	-	82.50	-
Step-Back Prompting	20.00	-	82.50	-
Self-Refine	22.00	-	84.00	-
Reflection	20.50	-	84.00	-
MAD	43.50	47.50	86.00	87.50
AgentPSO	36.00	52.50	84.50	89.00

updates reasoning skills instead of updating underlying medical knowledge, the evolved agents may explore more diverse solutions while still failing to form a reliable majority when the required clinical knowledge is not reliably represented. Thus, these results show that AgentPSO can improve answer coverage, as reflected by higher pass@k, but skill-only optimization is less effective for improving majority-vote accuracy in knowledge-intensive domains.

E Example of Results

E.1 Example of Velocity

This section provides a qualitative example of how the velocity evolves during AgentPSO optimization. We use Agent 3 as a representative example, which is initialized with the Self-Refine skill.

Figure 6 shows the velocity descriptions produced at different training iterations. Unlike the skill itself, the velocity represents the *update direction* that guides how the agent should revise its current reasoning behavior at each step. Each instruction in the velocity is annotated according to its main source of influence: **[d_i]** indicates content introduced by the feedback-guided self-reflective direction, **[GB]** indicates influence from the global-best skill, **[PB]** indicates influence from the agent's personal-best skill, **[s_i]** indicates influence from the agent's current skill, and **[prev_v]** indicates content carried over from the previous velocity. When the personal-best and current skills are identical, we use **[PGB]** because their contributions cannot be separated. These annotations illustrate how AgentPSO constructs each update direction by combining self-reflection feedback, guidance from the global-best and personal-best skill, the agent's own skill state, and accumulated momentum from previous velocities.

Overall, this example shows that the velocity in AgentPSO is not a simple copy of either the global-best or personal-best skill. Instead, it functions as a compositional update signal that synthesizes multiple information sources into a coherent revision direction. This behavior highlights the role of velocity as the mechanism that translates feedback and peer influence into gradual but meaningful agent evolution.

E.2 Example of Skill Evolution

This section provides a qualitative example of how an agent skill evolves during AgentPSO optimization. We again use Agent 3 as an example, which is initialized with the Self-Refine skill.

Figure 7 shows the detailed skill descriptions saved at different iterations. Each instruction is annotated according to its main source of influence: **[PB]** indicates instructions inherited from the

Table 15: **Qualitative trajectory of agent skill.** Across iterations, the skill evolves from generic self-refinement and output-format control toward structured problem-type classification, edge-case verification, proof-to-answer calibration, and strict benchmark-format compliance.

Iteration	Newly Emphasized Behavior
Iter. 1	Self-review, final-format validation, removal of extraneous output, and direct final-answer generation.
Iter. 3	Early problem-type classification, method selection, and explicit handling of symbolic, algebraic, geometric, and boundary-sensitive cases.
Iter. 9	Internal self-refinement, proof-to-answer calibration, domain/existence/uniqueness checks, and stricter benchmark-format compliance.

Table 16: **Functional analysis of accepted personal-best updates during skill evolution.** F, E, S, and G denote formatting/answer extraction, error correction/verification, strategy selection, and generic prompt optimization, respectively. Full AgentPSO yields more frequent and functionally diverse updates than the component-ablated variants.

Iter.	Full AgentPSO	w/o d_i	Personal Only	Global Only	Self-Reflection Only
1	F2 E2 S1 G1 (2)	F0 E1 S1 G1 (1)	F0 E1 S0 G1 (1)	–	–
2	F3 E3 S2 G0 (3)	–	F0 E0 S1 G1 (1)	–	F1 E1 S0 G1 (1)
3	F1 E1 S1 G0 (1)	F0 E1 S0 G1 (1)	–	F0 E1 S1 G1 (1)	–
4	F2 E2 S2 G0 (2)	–	–	–	F1 E1 S1 G0 (1)
5	–	–	–	–	–
6	F2 E2 S2 G1 (2)	F0 E1 S1 G1 (1)	–	–	–
7	F1 E0 S0 G0 (1)	–	–	F0 E1 S2 G2 (2)	–
8	–	F0 E1 S0 G1 (1)	–	–	–
9	F1 E1 S1 G1 (1)	F0 E1 S0 G1 (1)	–	F0 E0 S1 G1 (1)	–
10	–	–	–	–	–
Total	**F12 E11 S9 G3 (12)**	F0 E5 S2 G5 (5)	F0 E1 S1 G2 (2)	F0 E2 S4 G4 (4)	F2 E2 S1 G1 (2)

previous personal-best skill, **[GB]** indicates instructions influenced by the global-best skill, and **[d_i]** indicates instructions introduced by the feedback-guided self-reflective direction. These annotations help show how AgentPSO combines personal-best preservation, global-best guidance, and reflective feedback when updating an agent skill.

Table 15 provides a compact summary of the same trajectory. Across iterations, the skill evolves from a generic self-refinement instruction into a more structured reasoning policy. In the early stage, the agent mainly learns to align its response with the required answer format, such as checking the expected output type and removing extraneous content. In the middle stage, it begins to incorporate task-level reasoning strategies, including problem-type classification and method selection. By the later stage, the skill further emphasizes domain and edge-case verification, proof-to-answer calibration, and strict benchmark-format compliance. Overall, this example shows that AgentPSO does not simply replace an agent's initial prompt, but progressively refines it by integrating useful behaviors from multiple update sources.

E.3 Functional Analysis of Skill Evolution

Beyond the aggregate performance ablation in Table 6, we examine how different update components affect the functional content of the learned skills. For each DeepMath optimization run, we collect newly accepted personal-best updates at every iteration and use GPT-5.4-mini as an LLM judge to classify each update along four functional dimensions. Table 16 reports the functional composition of these accepted updates at each iteration. The value in parentheses denotes the number of newly accepted updates, while F/E/S/G indicate the corresponding functional behaviors:

- **F** (Formatting / Answer Extraction): Exact output form, answer-only responses, value-vs-expression distinctions, canonical notation, benchmark-specific format, final-answer extraction, or final-format compliance.

- **E** (Error Correction / Verification): Assumption checks, arithmetic / algebra checks, domain constraints, edge-case verification, branch / sign checks, existence/uniqueness checks, gap checks, or final correctness checks.
- **S** (Strategy Selection): Problem-type classification, governing-principle identification, method / template / theorem / algorithm selection, or planning before solving.
- **G** (Generic Prompt Optimization): Robust reasoning, clarity, concision, reliability, or general improvement language without concrete formatting, verification, or strategy-selection behavior.

Full AgentPSO produces the most accepted updates (12 in total) and the broadest functional coverage across formatting, verification, and strategy selection. Removing the self-reflective direction d_i reduces the number of accepted updates to five and yields more limited functional changes. The component-only variants further show complementary roles: self-reflection provides targeted corrective behaviors, while personal-best and global-best guidance help preserve and propagate useful reasoning behaviors. Together with the quantitative ablation in Table 6, these results suggest that self-reflective direction d_i provides targeted corrective signals, while personal-best and global-best guidance help preserve and propagate useful behaviors. Their combination in the full AgentPSO update leads to more frequent and functionally diverse skill evolution.

F PROMPTS IN AGENTPSO

Figures 8–11 summarize the prompts used in AgentPSO for the mathematical reasoning tasks. The pipeline first prompts each agent to solve the problem using its current skill, then uses the agents' outputs to derive a self-reflective update direction. This direction is combined with the previous velocity, personal-best skill, and global-best skill to generate a PSO-guided natural-language velocity. The generated velocity is then applied to the current skill to produce the next skill while preserving the agent's identity.

Velocity at Iteration 1

[PGB] [d_i] Revise the workflow to solve first, then run a stricter final validation pass. [PGB] [d_i] Keep the current review habit, but make the last check more deliberate about benchmark alignment and answer exactness. [d_i] Confirm whether the task expects an expression, equation, or plain value. [d_i] Use the canonical symbol/notation. [d_i] Remove any stray wording, braces, JSON, or metadata before responding. [d_i] Preserve the strong mathematical reasoning. [d_i] When the answer is simple, prefer a direct, problem-specific argument over a looser derivation. [d_i] In symbolic or boundarysensitive problems, especially algebra/cardinality and geometry, explicitly verify the final form and edge cases. [d_i] **[GB]** Ensure that the response matches the required format exactly.

Velocity at Iteration 3

[PGB] [prev_v] [d_i] Revise as Self-Refine with a stricter final validation pass. [PGB] Solve first, then re-check the answer. [d_i] [PGB] [prev_v] Check the answer against the exact required output type and benchmark notation. [PGB] Keep the self-correcting review habit. [PGB] [d_i] Verify whether the response must be a plain value, expression, or equation. [d_i] [PGB] [prev_v] Strip all commentary, JSON, braces, metadata, and unsupported equalities unless explicitly requested. [d_i] For choice or parameter problems, return only the required symbol or scalar. [PGB] [prev_v] For symbolic, algebraic, geometric, or boundary-sensitive tasks, preserve careful reasoning internally but output only the canonical final form. [PGB] [prev_v] [d_i] Check edge cases, existence/domain conditions, and notation consistency. [PGB] Start by classifying the problem type and choosing the most fitting method. [d_i] [PGB] [prev_v] Perform a brief final check for correctness, minimality, and exact token match before responding.

Velocity at Iteration 9

[s_i] **[GB]** Revise as Self-Refine with a stronger step-back start. **[GB]** [s_i] [prev_v] First classify the task, identify the governing principle, and determine the exact required output form before solving. **[GB]** [s_i] Use the standard method for the problem class. [s_i] **[GB]** [prev_v] Keep all reasoning internal. [s_i] **[PB]** Solve directly, then run a strict self-check. [s_i] **[PB]** [prev_v] Verify correctness, edge cases, domain/existence/uniqueness, and boundary conditions when relevant. [d_i] Tighten proof-to-answer calibration. [d_i] Pay special attention to abstract algebra/field-theory and quotient/classification problems. [d_i] Verify claims against the exact prompt and avoid overgeneralizing from partial structure. **[GB]** [s_i] **[PB]** [prev_v] Strengthen final-form control so the response matches the benchmark's expected shape exactly. **[PB]** **[GB]** [prev_v] Distinguish whether the final answer should be a value, equation, or canonical expression. **[PB]** [s_i] **[GB]** [prev_v] Use minimal notation, with no extra commentary, proof, metadata, or unsupported equalities. [d_i] Add a final consistency pass for sign, counting, and maximal/subquotient-style arguments. [d_i] [s_i] [prev_v] Ensure symbolic or yes/no prompts return only the requested form.

Figure 6: **Example of velocity across AgentPSO training iterations.** The velocity is evolved from an empty string. **[PB]** denotes influence from Agent's personal-best skill, **[GB]** denotes influence from the global-best skill, [d_i] denotes the self-reflective direction at the corresponding iteration, [s_i] denotes influence from Agent's current skill, and [prev_v] denotes accumulated influence from the previous velocity. When the personal-best and current skills are identical, we use [PGB] because their contributions cannot be separated.

Skill at Iteration 1

- **[PB]** Solve the problem first, then review and refine the solution.
- **[PB]** Keep the reasoning strong and self-correcting.
- **[GB]** **[d_i]** In the final pass, validate strict alignment with the task's required output format.
- **[d_i]** Check whether the answer should be an expression, equation, or plain value, and use the canonical notation/symbols.
- **[d_i]** Remove any stray wording, braces, JSON, metadata, or other extraneous content before responding.
- **[PB]** When the solution is simple, give a direct, problem-specific answer rather than an unnecessarily long derivation.
- **[d_i]** For symbolic, boundary-sensitive, or geometry/algebra problems, explicitly verify edge cases and the final form.
- **[GB]** **[d_i]** Return only the final answer in the exact form the task expects.

Skill at Iteration 3

- **[PB]** Solve the problem first, then self-review and refine the result.
- **[PB]** Preserve a disciplined, self-correcting approach aligned with the task.
- **[GB]** Classify the problem type early and choose the most appropriate method.
- **[PB]** **[d_i]** In the final pass, verify the exact required output type: plain value, expression, or equation.
- **[GB]** **[d_i]** Use canonical notation and the benchmark's expected symbols; ensure domain, existence, and edge-case conditions are satisfied.
- **[PB]** For simple tasks, answer directly without extra derivation.
- **[d_i]** For symbolic, algebraic, geometric, or boundary-sensitive tasks, check edge cases and confirm the final form is correct.
- **[PB]** **[d_i]** Remove commentary, metadata, braces, JSON, and unsupported equalities unless explicitly requested.
- **[d_i]** For choice or parameter problems, return only the required symbol or scalar.
- **[PB]** **[GB]** Output only the final answer in the exact form requested.

Skill at Iteration 9

- **[PB]** Self-Refine: solve internally, then self-check and refine before responding.
- **[GB]** First classify the task, identify the governing principle, and determine the exact required output form.
- **[GB]** Use the standard method for the problem class and reason carefully, but keep all reasoning internal.
- **[PB]** **[d_i]** Solve directly, then verify domain, existence, uniqueness, edge cases, boundary conditions, and any relevant sign/counting/maximality issues.
- **[d_i]** Tighten proof-to-answer calibration: make sure the result matches the exact prompt and do not overgeneralize from partial structure, especially in abstract algebra, field theory, and quotient/classification problems.
- **[PB]** Perform a strict final validation for correctness, completeness, and format compliance.
- **[GB]** Prefer canonical notation and the simplest correct final expression.
- **[GB]** Match the benchmark's expected shape exactly—value, equation, or canonical form—with minimal notation.
- **[GB]** Output only the final answer, with no extra commentary, scaffolding, proof, metadata, or unsupported equalities.
- **[d_i]** For yes/no or symbolic prompts, return only the requested form.

Figure 7: **Example of skill evolution across AgentPSO training iterations.** The skill is evolved from "*First solve the problem, then review and improve the solution.*" **[PB]** denotes instructions inherited from the previous personal-best skill, **[GB]** denotes instructions influenced by the global-best skill, and **[d_i]** denotes instructions introduced by the feedback-guided reflective direction.

Problem Solving Prompt

You are a mathematics competition agent.
Base rules:
- Solve the provided MATH problem.
- The answer may be a number, expression, tuple, interval, or simplified LaTeX expression.
- Do not invent missing problem details.
- Follow the current skill file below.

Current skill file:
{skill_text}

Solve this problem using only your current skill.

Problem:
{question}

Return exactly one JSON object:
{
"agent_id": {agent_id},
"reasoning": "...",
"answer": "..."
}

Figure 8: **Problem-solving prompt used by each agent.** Given the current skill file, each agent independently solves the input problem and returns its reasoning trace and final answer in JSON format.

Generate Self-reflective Direction Eq. (1)

Current agent skill:
{current_skill}

Agent's own reasoning traces and answers:
{own_outputs_json}

Other agents' reasoning traces and answers, including correctness:
{peer_outputs_json}

Instruction:
Analyze the agent's performance compared with peers.
Identify general reasoning improvements.
Do not overfit to a single problem.
Do not rewrite the skill yet.
Return only the update direction.

Figure 9: **Self-reflective direction generation prompt.** This prompt generates an update direction by comparing the current agent's performance with peer agents, aiming to identify general reasoning improvements.

PSO-Guided Velocity Eq. (2)

Agent identity to preserve:
{agent_identity}

Previous velocity v_i:
{previous_velocity}

Self-reflective direction d_i:
{direction}

Current skill s_i:
{current_skill}

Personal best skill p_i:
{personal_best_skill}

Global best skill g:
{global_best_skill}

Instruction:
Combine the previous velocity, self-reflective direction, lessons from the personal best, and lessons from the global best.
Focus on generalizable improvements.
Do not copy the personal best or global best directly.
Preserve the agent's identity.
Return a concise natural-language velocity of at most {max_velocity_words} words.

Figure 10: **Prompt for generating the PSO-guided velocity v_i^{t+1}.** Given the previous velocity, the self-reflective direction, the personal-best skill, and the global-best skill, the model generates a concise natural-language velocity that captures generalizable improvements while preserving the agent's identity.

Skill Update from Velocity Eq. (3)

Agent identity to preserve:
{agent_identity}

Current skill s_i:
{current_skill}

Velocity v_i:
{velocity}

Instruction:
Rewrite the skill according to the velocity.
Keep the skill concise and general.
Preserve the agent's original role.
Remove redundant, overly specific, or contradictory instructions.
Use at most 10 bullet points and at most {max_skill_words} words.
Do not copy another skill verbatim.
Return only the updated skill.

Figure 11: **Prompt for applying the velocity to the skill.** Given the current skill and the PSO-guided velocity, the model rewrites the skill into an updated form that preserves the agent's identity while incorporating concise and generalizable improvements.