Agent-GWO: Collaborative Agents for Dynamic Prompt Optimization in Large Language Models

Xudong Wang[1], **Chaoning Zhang**[2*], **Chenghao Li**[2], **Shuxu Chen**[1], **Qigan Sun**[1], **Jiaquan Zhang**[2], **Fachrina Dewi Puspitasari**[2], **Tae-Ho Kim**[3], **Jiwei Wei**[2], **Malu Zhang**[2], **Guoqing Wang**[2], **Yang Yang**[2], and **Heng Tao Shen**[4]

[1]Kyung Hee University [2]University of Electronic Science and Technology of China
[3]Nota Inc. [4] Tongji University
wangcurry33@gmail.com ✉chaoningzhang1990@gmail.com

Abstract

Large Language Models (LLMs) have demonstrated strong capabilities in complex reasoning tasks, while recent prompting strategies such as Chain-of-Thought (CoT) have further elevated their performance in handling complex logical problems. Despite these advances, high-quality reasoning remains heavily reliant on manual static prompts and is sensitive to decoding configurations and task distributions, leading to performance fluctuations and limited transferability. Existing automatic prompt optimization methods typically adopt single-agent local search, failing to simultaneously optimize prompts and decoding hyperparameters within a unified framework to achieve stable global improvements. To address this limitation, we propose Agent-GWO, a dynamic prompt optimization framework for complex reasoning. Specifically, we unify prompt templates and decoding hyperparameters as inheritable agent configurations. By leveraging the leader-follower mechanism of the Grey Wolf Optimizer (GWO), we automatically select three leader agents (α, β, and δ) to guide the collaborative updates of the remaining agents, enabling iterative convergence toward robust optimal reasoning configurations that can be seamlessly integrated for inference. Extensive experiments on multiple mathematical and hybrid reasoning benchmarks across diverse LLM backbones show that Agent-GWO consistently improves accuracy and stability over existing prompt optimization methods.

1 Introduction

LLMs have recently achieved substantial progress across a range of reasoning tasks, particularly when leveraged with prompting strategies such as Chain-of-Thought (CoT) that elicit step-by-step reasoning (Wei et al., 2022; Kojima et al., 2022). Despite these gains, strong average performance does not necessarily translate into reliable reasoning behavior. As problems become longer and more complex, accuracy often degrades significantly (Cao et al., 2024; Li et al., 2025a). More critically, performance remains highly sensitive to prompt wording, exemplar ordering, and contextual perturbations: minor edits or re-orderings that leave the underlying task unchanged can nonetheless precipitate large swings in performance (Chatterjee et al., 2024; Ariyani et al., 2025; Madani et al., 2026). This prompt's fragility makes it difficult to distinguish genuinely robust reasoning steps from those that are merely accidentally correct. Furthermore, early deviations in the reasoning chain can cascade through subsequent steps, severely compromising the reliability of multi-step solutions (Yue et al., 2023; He et al., 2025; Wang et al., 2025).

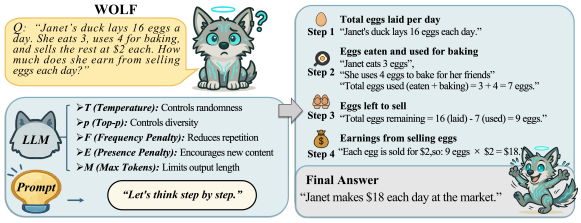

Figure 1: GWO abstracts each LLM agent as a "wolf" which is characterized by hyperparameters & reasoning prompt. These two guide the behavior of the LLMs during the optimization process.

To address the above reliability challenge, prior work has proposed a range of inference-time enhancement strategies that improve the robustness of single-pass decoding primarily through trajectory generation and search, such as self-consistency sampling, multi-path exploration, and tree/graph-structured reasoning (Madaan et al., 2023; Zhang et al., 2024b; Teng et al., 2025; Zhang et al., 2026a,d; Huang et al., 2026). Although these methods often deliver gains without additional training, they are typically built upon manually crafted static prompts. As a result, performance can still

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 16648–16670
July 2-7, 2026

fluctuate markedly under task transfer, contextual perturbations, or minor prompt rewrites, leaving prompt brittleness and poor reusability largely unresolved(Jin et al., 2024). Moreover, repeated sampling and structured search commonly introduce extra inference cost and system complexity, which limit their applicability in resource-constrained or low-latency settings (Yue et al., 2023; He et al., 2025). In contrast, training-based adaptation can offer more systematic improvements, but usually requires extra data, training iterations, and engineering effort, making it less suitable as a lightweight, general-purpose inference-stage solution (Lester et al., 2021; Hu et al., 2022). Overall, existing work largely improves reasoning trajectories under fixed prompts, rather than providing a reproducible and scalable mechanism that can automatically optimize both the prompting strategy and the decoding behavior at inference time.

Motivated by this gap, we ask a key question: ***at inference time, can we reduce reliance on manual prompt trial-and-error and automatically discover prompt-configuration pairs that are more stable and better aligned with task requirements?*** Our core observation is that, for a given frozen model, reasoning quality is mainly governed by two controllable factors: the prompt template (how the reasoning process is organized and constrained) and the decoding configuration (e.g., stochasticity, repetition penalties, and length budgets). Systematically exploring this joint space via population-based search offers the potential to uncover reasoning strategies that are both more stable and more task-adaptive.

To this end, we propose Agent-GWO, a dynamic prompt optimization framework for complex reasoning. As illustrated in Figure 1, each agent is defined by a reasoning prompt template together with a decoding configuration, and a population of such agents is optimized collaboratively in an iterative manner. At each iteration, multiple agents generate reasoning traces in parallel and are ranked according to task-level performance. We then design a hierarchical collaborative update mechanism that enables structured information sharing among agents. By exploiting leader–follower dynamics inspired by the Grey Wolf Optimizer, this mechanism allows top-performing agents to guide population-level updates and steer the evolution of the remaining agents. This design preserves population diversity to encourage exploration while ensuring stable and controllable convergence toward high-quality reasoning configurations. The resulting optimized configuration can be directly reused at inference time without any additional training.

Our contributions are summarized as follows:

- We propose an automated inference-time adaptation paradigm that jointly optimizes prompt templates and decoding parameters at test time, improving both accuracy and stability without additional training.
- We design a hierarchical leader–follower swarm mechanism inspired by GWO to stabilize multi-agent prompt search and reduce sensitivity to prompt variants.
- Extensive experiments on mathematical and hybrid reasoning benchmarks across diverse LLM backbones show consistent gains under controllable inference budgets over strong manual baselines and prior prompt optimization methods.

2 Related Work

2.1 Prompt Engineering and Static Chain-of-Thought Methods

Prompting is a widely used approach for adapting LLMs in zero- and few-shot settings (Brown et al., 2020). Chain-of-Thought (CoT) prompting (Wei et al., 2022) elicits step-by-step reasoning and has inspired extensions such as self-consistency (Wang et al., 2022), least-to-most decomposition (Zhou et al., 2022), and structured variants including Self-Ask (Press et al., 2022), Program-of-Thoughts (Chen et al., 2022), and Graph-of-Thoughts (Liu et al., 2023).

Despite their success, these methods still rely heavily on task-specific prompt design (Zhang et al., 2022) and can be sensitive to prompt phrasing and exemplar ordering (Lu et al., 2022). Moreover, long reasoning traces and multi-sample strategies often increase inference cost (Zhou et al., 2022; Zheng et al., 2025; Cao et al., 2026; Zheng et al., 2026b; Zhang et al., 2026c). Recent work therefore studies automatic prompt optimization to reduce manual trial-and-error and improve transferability, e.g., via evolutionary or gradient-free search and feedback-driven refinement. Our work follows this direction but focuses on population-based, leader-guided optimization over both prompt templates and decoding configurations under frozen model weights.

2.2 Collaborative Reasoning with Multi-Agent LLMs

Multi-Agent Systems (MAS) provide a natural framework for collaborative reasoning through coordination and role specialization (Wooldridge and Jennings, 1995; Ferber and Weiss, 1999). With LLMs, MAS has been used for planning and reasoning via agent communication and iterative interaction (Naveed et al., 2023; Kumar, 2024; Xi et al., 2025), with representative frameworks such as AutoGen (Wu et al., 2023) and Chain of Agents (Zhang et al., 2024c). Prior work explores modular agent designs (Liu et al., 2025; Zhang et al., 2026b), debate-style aggregation (Du et al., 2023; Xu et al., 2026), and communication/coordination protocols (Fioretto et al., 2018; Fipa, 2002; Jiang and Lu, 2018; Yang et al., 2026b,a), and applies them to diverse domains (Park et al., 2023; Zhang et al., 2024a; Lim et al., 2024; Li et al., 2023; Zheng et al., 2026a; Wang et al., 2026).

Complementary to this line, we cast multi-agent collaboration as an explicit iterative optimization process, where strong agents guide the update of others to improve task-specific prompting and decoding behaviors.

3 Method

3.1 Theoretical Background and Problem Formulation

Grey Wolf Optimizer (GWO). GWO (Mirjalili et al., 2014) is a population-based metaheuristic that maintains N candidate solutions ("wolves") and updates them using a leader–follower hierarchy. At each iteration, the top three solutions are treated as leaders (α, β, δ), and other wolves (ω) are guided toward them. For a wolf $\mathbf{X}_i$ in a D-dimensional space, the standard GWO update averages the three leader-guided positions:

$$\mathbf{X}_i(t+1) = \frac{\mathbf{X}_i^{(\alpha)}(t+1)+\mathbf{X}_i^{(\beta)}(t+1)+\mathbf{X}_i^{(\delta)}(t+1)}{3} \quad (1)$$

where $\mathbf{X}_i^j(t+1)$ denotes the leader-guided update induced by leader $j \in \{\alpha, \beta, \delta\}$ using the encircling rule (Appendix A.1).

Agent Structure and Definition. Suppose there are n agents. The j-th agent, denoted as Agent_j, consists of a large language model LLM_j and a prompt template prompt_j, i.e., $\text{Agent}_j = \{\text{LLM}_j, \text{prompt}_j\}$. The language model LLM_j contains a shared model parameter set $\boldsymbol{\theta}$ and an agent-specific hyperparameter set $\boldsymbol{\eta}_j = \{T_j, p_j, F_j, E_j, M_j\}$, which represent temperature, top-p threshold, frequency penalty, presence penalty, and maximum token length, respectively. Therefore, $\text{LLM}_j = \{\boldsymbol{\eta}_j, \boldsymbol{\theta}\}$, and $\text{Agent}_j = \{\{\boldsymbol{\eta}_j, \boldsymbol{\theta}\}, \text{prompt}_j\}$.

Hyperparameter Sampling. To ensure diversity and stability in agent behavior, we design a hyperparameter sampling strategy for the set $\boldsymbol{\eta}_j = \{t_j, p_j, f_j, e_j, m_j\}$, where a clipping function $\text{clip}(x, [a, b]) = \max(a, \min(x, b))$ constrains sampled values within valid ranges, thereby maintaining controlled and consistent generation. Specifically, the temperature t_j and the top-p threshold p_j are independently drawn from normal distributions $\mathcal{N}(\mu_t, \sigma_t^2)$ and $\mathcal{N}(\mu_p, \sigma_p^2)$, then clipped to intervals $[a_t, b_t]$ and $[a_p, b_p]$, respectively. These parameters regulate the generation distribution: higher values enhance creativity and diversity, while lower values promote predictability and stability. The frequency penalty f_j, sampled from $\mathcal{N}(\mu_f, \sigma_f^2)$ and clipped to $[a_f, b_f]$, mitigates repetitiveness by penalizing tokens proportionally to their prior frequency. The presence penalty e_j, sampled from $\mathcal{N}(\mu_e, \sigma_e^2)$ and clipped to $[a_e, b_e]$, encourages novelty by uniformly penalizing any previously generated token, thus fostering the introduction of new words or concepts. Finally, the maximum token length m_j is set to a fixed constant c_m for tasks requiring predetermined length, while for adaptive-length tasks, it is uniformly drawn from a discrete set $\mathcal{M}$, i.e., $m_j \sim \text{Uniform}(\mathcal{M})$.

This sampling approach uses normal or uniform distributions to introduce controlled variability, while clipping ensures hyperparameters remain within reasonable bounds, optimizing creativity, stability, and content quality.

Problem Formulation. We denote the reasoning task dataset as $\mathcal{D} = \{q_1, q_2, \ldots, q_N\}$, where each element q_i represents a specific question. The dataset consists of N question samples in total. During the reasoning process, given any question $q \in \mathcal{D}$, a specific agent, denoted as Agent_j, processes the question and outputs two components: first, a detailed chain of thought (denoted as CoT_j), which illustrates the step-by-step reasoning process taken by the agent; and second, a final answer (denoted as Answer_j) that the agent derives based on the chain of thought. This reasoning process can be formally expressed as a function:

$$f(\text{Agent}_j, q) = (\text{CoT}_j, \text{Answer}_j), \quad q \in \mathcal{D} \quad (2)$$

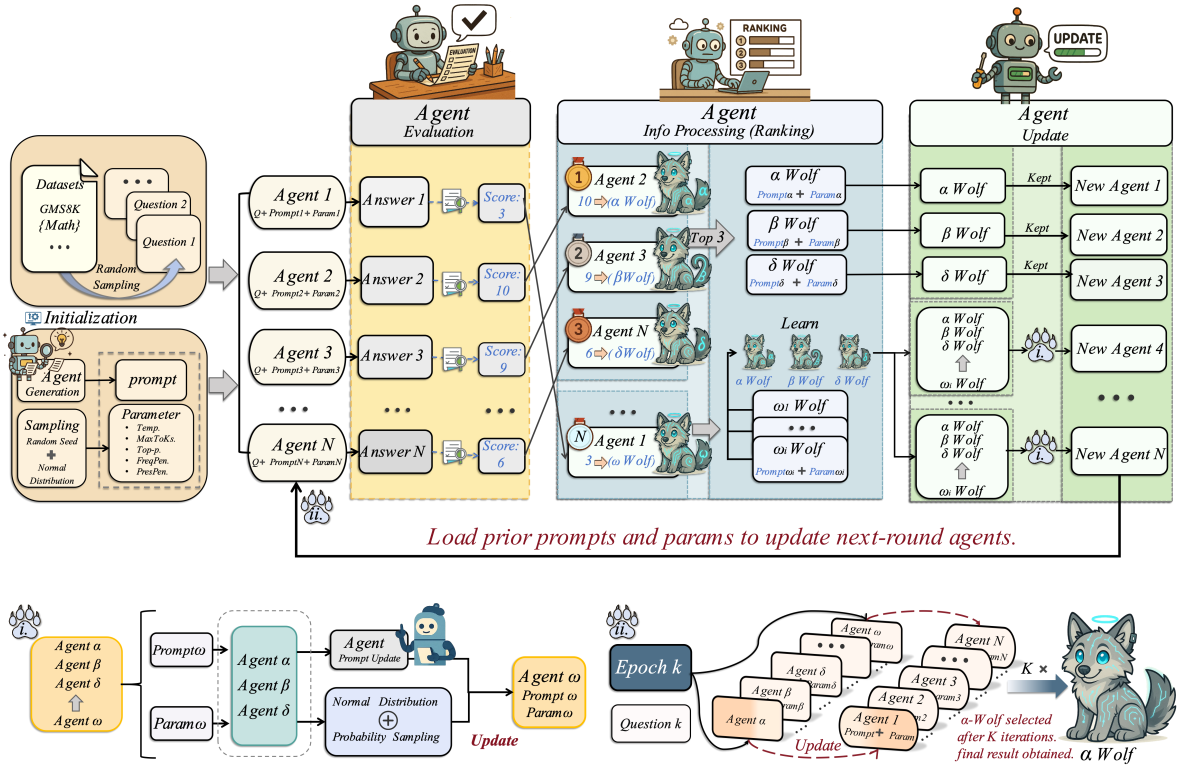

Figure 2: The overall Agent-GWO framework operates by having each agent process a dataset, Prompt, and parameters to produce outputs, which are then scored. Only the top-3 agents (α, β, and δ) are differentiated, while the rest (i.e., ω) are updated using the prompt and parameters from the previous round. This iterative ranking and update process continues until termination, after which the final agent is set as the top-ranked α from the last iteration. **Note:** The left-bottom subfigure illustrates the learning process of each agent at each iteration and the right-bottom subfigure illustrates the optimization process.

Evaluation Function. For verifiable reasoning benchmarks with gold labels, we define the fitness of agent j as the exact-match accuracy on a validation batch D_{val}:

$$\text{FITNESS}_j = \frac{1}{|D_{\text{val}}|} \sum_{q \in D_{\text{val}}} \mathbb{1}[\text{Answer}_j(q) = y(q)] \,. \tag{3}$$

To stabilize leader selection, we compute auxiliary scores $\left(s^{\text{logic}}, s^{\text{creativity}}, s^{\text{complete}}\right)$ using a fixed LLM-judge prompt (Appendix A.2). These auxiliary scores are not used to replace accuracy, but only to rank agents when accuracies are tied. For non-verifiable tasks, we use the same judge to define

$$\text{FITNESS}_j = w_1 s^{\text{logic}} + w_2 s^{\text{creativity}} + w_3 s^{\text{complete}}, \tag{4}$$

where the weights are calibrated on 200 human-labeled samples. We set the calibrated weights to $(w_1 = 0.5, w_2 = 0.2, w_3 = 0.3)$, reflecting the higher importance of logical correctness. All judge-based scores are averaged over three independent judge seeds.

3.2 Multi-Agent Grey Wolf Optimizer Framework

As illustrated in Figure 2, we adapt the GWO to optimize LLM agent configurations by modeling each agent as a "wolf" in a population. Each agent $\text{Agent}_j = (\boldsymbol{\eta}_j, \text{prompt}_j)$ corresponds to a candidate solution, defined by its decoding hyperparameters and prompt template. The overall procedure follows GWO's leader–follower hierarchy, where the top-performing agents (α, β, δ) guide the updates of the remaining agents over multiple iterations. The full pseudocode is provided in Appendix A.3.

Initialization. The algorithm initializes a population $\mathcal{A}$ of n agents. For each agent Agent_j, hyperparameters $\boldsymbol{\eta}_j$ are randomly sampled from predefined ranges (e.g., temperature $\in [0, 1]$, top-p $\in [0, 1]$) using a normal distribution. Similarly, the prompt prompt_j is sampled from a set of predefined CoT

prompt templates or generated via a template generation function. This diverse initialization ensures broad exploration of the search space.

Optimization Loop. The optimization phase iterates K times, with each iteration evaluating and updating the agent population. For each agent $Agent_j$, a question q is sampled from $\mathcal{D}$, and the LLM generates a CoT trace and answer using the agent's configuration. The fitness of the answer is computed using a carefully designed evaluation function, which assesses the response based on three crucial dimensions: the logical consistency of the reasoning process, the ingenuity of the reasoning approach, and the comprehensiveness of the reasoning content. This multi-dimensional evaluation ensures a thorough and accurate assessment of the answer's quality. The top three agents, denoted α, β, and δ, are selected based on their fitness scores, representing the best, second-best, and third-best solutions, respectively.

Non-elite agents ($\mathcal{A} \setminus \{\alpha, \beta, \delta\}$) are updated to converge toward the top performers. For each hyperparameter $\eta_j^{(k)}$ in agent $Agent_j$, new values are sampled from normal distributions centered at the top agents: $X_r \sim \mathcal{N}(\eta_r^{(k)}, \sigma^2)$, for $r \in \{\alpha, \beta, \delta\}$, and combined as a weighted average: $\eta_j^{(k)} = w_\alpha X_\alpha + w_\beta X_\beta + w_\delta X_\delta$, with weights satisfying $w_\alpha > w_\beta > w_\delta$ and $w_\alpha + w_\beta + w_\delta = 1$. This prioritizes the influence of the best-performing agent (α) while maintaining diversity. The prompt is adapted by a structured LLM-driven update function $\mathrm{PromptAdaptation}(\cdot)$. Formally, $P_j^{(k+1)} = \mathrm{PromptAdaptation}(P_j^{(k)}, P_\alpha, P_\beta, P_\delta, \mathcal{I})$, where $P_j^{(k)}$ denotes the current prompt of agent j, $P_\alpha, P_\beta, P_\delta$ are the prompts of the top-three elite agents, and $\mathcal{I}$ is a fixed system instruction. Under $\mathcal{I}$, the LLM performs lightweight edits on $P_j^{(k)}$ while preserving the task objective and evaluation criteria. The allowed edits are limited to step re-ordering, semantically equivalent paraphrasing, and formatting or constraint adjustments.

Overall procedure. Each iteration consists of generation, evaluation, ranking, and update, repeated for K rounds until returning the final leader configuration.

4 Experiments

In this section, we conduct a systematic evaluation of the proposed Agent-GWO framework to verify its effectiveness on complex reasoning tasks. Our evaluation covers the reasoning performance (Sec 4.2), adaptability performance to CoT (Sec 4.3), and ablation study (Sec 4.5).

4.1 Experimental Setup

4.1.1 Datasets

We evaluate the proposed Agent-GWO framework on a comprehensive suite of 11 benchmarks spanning two complementary categories: mathematical reasoning and hybrid reasoning. The mathematical reasoning set includes GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), MultiArith (Roy and Roth, 2015), and ASDiv (Miao et al., 2020). The hybrid reasoning set includes AQUA (Ling et al., 2017), MMLU (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), Date (Suzgun et al., 2023), CLUTRR (Sinha et al., 2019), and MATH_MIX. Collectively, these benchmarks span numerical computation (arithmetic, algebra, geometry, and combinatorics) and broader reasoning skills, including logical inference, knowledge/commonsense recall, temporal reasoning, and multi-step relational chaining. This diversity provides a robust testbed for evaluating GWO on multi-hop, cross-domain, and constraint-heavy reasoning tasks.

4.1.2 Baselines

We compare Agent-GWO with seven representative reasoning-enhancement baselines that can be evaluated under the same backbone setting for a fair comparison: Chain-of-Thought prompting (CoT) (Wei et al., 2022), its self-consistency variant CoT-SC with n=5 samples (Wang et al., 2022), iterative self-feedback refinement (Self-Refine) (Madaan et al., 2023), the feedback-driven workflow method AFlow (Zhang et al., 2024b), and structured reasoning paradigms including Tree-of-Thought (ToT) (Yao et al., 2023), Graph-of-Thought (GoT) (Besta et al., 2024), and Atom-of-Thought (AoT) (Teng et al., 2025). To assess robustness and generalization across model families and scales, we conduct experiments with five LLM backbones: GPT-4o-mini, GPT-4.1-mini, GPT-4.1-nano, Qwen2.5-Coder-7B-Instruct, and Gemma-3-12b-it.

4.1.3 Implementation Details

Following established practices in workflow optimization and inference-time search (Zhang et al., 2024b; Saad-Falcon et al., 2024; Hu et al., 2024), we adopt a strict optimization–evaluation separa-

Model	Math Reasoning Tasks						Hybrid Reasoning Tasks					Avg
	GSM8K	MATH	SVAMP	MultiArith	ASDiv	AQUA	MMLU	BBH	Date	CLUTRR	MATH_MIX	
GPT-4o-mini												
CoT	85.4%	74.8%	84.7%	89.5%	92.3%	65.3%	63.4%	66.6%	52.1%	66.2%	80.1%	74.6%
CoT-SC/n=5	89.9%	76.3%	85.8%	89.7%	93.3%	70.7%	67.1%	69.0%	54.7%	72.2%	81.6%	77.3%
Self-Refine	88.9%	75.1%	85.0%	89.9%	92.2%	67.8%	64.8%	67.5%	53.0%	68.4%	83.8%	76.0%
AFlow	93.5%	78.9%	88.7%	91.9%	93.4%	72.0%	69.5%	68.0%	66.0%	72.8%	85.3%	80.0%
ToT	94.9%	77.8%	89.6%	89.8%	93.6%	72.4%	71.6%	69.9%	69.2%	73.6%	85.6%	80.7%
GoT	93.2%	78.6%	89.2%	91.0%	94.2%	73.3%	71.3%	70.1%	65.2%	74.2%	84.4%	80.4%
AoT	95.0%	**83.6%**	91.5%	92.6%	94.1%	75.1%	74.7%	72.4%	74.7%	75.2%	86.9%	83.3%
Agent-GWO	**95.9%**	80.2%	**92.3%**	**95.3%**	**95.5%**	**75.9%**	**75.3%**	**73.9%**	**75.8%**	**76.4%**	**87.5%**	**84.0%**
Qwen2.5-Coder-7B-Instruct												
CoT	77.5%	65.8%	82.7%	84.6%	87.3%	60.9%	55.4%	47.4%	31.3%	20.4%	70.4%	62.2%
CoT-SC/n=5	80.0%	67.3%	83.9%	86.8%	89.8%	62.0%	56.1%	49.1%	32.7%	20.8%	73.9%	63.9%
Self-Refine	79.0%	70.4%	83.3%	86.4%	87.6%	61.2%	55.8%	48.6%	31.9%	20.8%	74.6%	63.6%
AFlow	84.0%	71.9%	86.0%	86.8%	90.1%	61.8%	56.5%	52.5%	34.0%	22.7%	77.0%	65.8%
ToT	83.6%	71.3%	85.6%	83.0%	89.8%	61.5%	56.9%	55.0%	34.5%	23.0%	77.8%	65.6%
GoT	84.8%	71.7%	87.5%	85.1%	92.5%	62.9%	56.8%	53.4%	34.7%	23.8%	76.8%	66.4%
AoT	88.5%	72.0%	90.0%	88.4%	**94.1%**	63.0%	56.8%	57.0%	36.2%	26.4%	78.8%	68.3%
Agent-GWO	**89.1%**	**74.1%**	**90.1%**	**93.3%**	93.3%	**63.3%**	**58.3%**	**58.8%**	**37.1%**	**27.8%**	**83.5%**	**69.9%**
Gemma-3-12b-it												
CoT	83.5%	72.8%	79.3%	82.7%	91.0%	69.2%	68.4%	64.9%	78.0%	49.3%	75.1%	74.0%
CoT-SC/n=5	85.9%	74.6%	80.8%	85.1%	93.2%	71.5%	70.4%	66.5%	80.0%	52.0%	77.0%	76.1%
Self-Refine	84.6%	77.6%	80.3%	87.5%	89.2%	70.2%	69.1%	65.2%	79.2%	50.6%	79.6%	75.7%
AFlow	89.0%	79.0%	85.2%	89.0%	92.2%	73.8%	70.8%	66.0%	81.3%	53.0%	82.6%	78.4%
ToT	88.4%	79.5%	83.9%	88.2%	91.8%	72.6%	70.6%	68.4%	81.0%	52.6%	83.5%	78.2%
GoT	89.2%	80.3%	86.8%	86.8%	93.9%	74.5%	71.6%	68.0%	81.5%	53.7%	82.4%	79.0%
AoT	91.4%	79.6%	**91.8%**	87.5%	93.5%	76.7%	71.9%	68.7%	82.0%	54.8%	84.0%	80.2%
Agent-GWO	**92.8%**	**82.1%**	90.9%	**95.9%**	**94.2%**	**78.5%**	**72.7%**	**70.4%**	**84.5%**	**56.9%**	**88.0%**	**82.4%**

Table 1: Performance comparison of Agent-GWO against baselines across math and hybrid reasoning tasks. Bold indicates the best result, and underlined indicates the second-best result.

tion protocol. For benchmarks with official splits, including GSM8K, SVAMP, MultiArith, ASDiv, AQUA, MMLU, and CLUTRR, optimization is performed exclusively on the training/dev side to compute exact-match fitness and update prompt–configuration pairs, while the test split is used only for the final evaluation. For MATH, we follow (Hong et al., 2025) and evaluate on 617 Level-5 problems across four representative categories; optimization follows the same protocol using labeled data from the official training side, ensuring that no test labels are used for feedback. For benchmarks without official train/dev splits, including BBH, Date, and MATH_MIX, we follow prior workflow-optimization practice (Zhang et al., 2024b; Saad-Falcon et al., 2024; Hu et al., 2024) and perform a fixed-seed random hold-out split, where the validation and test portions are constructed in a 1:4 ratio. The validation portion serves as the optimization pool for fitness computation, while the test portion is strictly held out for final evaluation.

Unless otherwise specified, we use the default GWO configuration with $n = 5$ agents and $K = 10$ optimization iterations. This setting provides a practical trade-off between computational budget and performance gains, and is applied consistently across datasets and backbones to ensure fair comparison. Empirical support for this configuration is provided in the ablation study (Section 4.5).

4.2 Performance Evaluation

Table 1 reports a systematic comparison of Agent-GWO against seven representative reasoning-enhancement baselines (CoT, CoT-SC, Self-Refine, AFlow, ToT, GoT, and AoT) across eleven benchmarks under three LLM backbones. Overall, Agent-GWO achieves the best average performance (Avg) on all backbones, and attains the best or second-best results on most individual tasks, demonstrating consistent stability and strong competitiveness across settings. In contrast, methods such as CoT and Self-Refine yield more limited overall gains on complex mathematical and hybrid reasoning benchmarks, and their effectiveness is more sensitive to the specific task and backbone configuration, leading to larger performance fluctuations. These results also highlight the benefit of Agent-GWO's collaborative optimization: by

Model / Method	Math Reasoning Tasks						Hybrid Reasoning Tasks					Avg. Increase
	GSM8K	MATH	SVAMP	MultiArith	ASDiv	AQUA	MMLU	BBH	Date	CLUTRR	MATH_MIX	
GPT-4o-mini												
CoT	85.4%	74.8%	84.7%	89.5%	92.3%	65.3%	63.4%	66.6%	52.1%	66.2%	80.1%	-
CoT+GWO/n=5	**95.1%**	**79.9%**	**92.4%**	**99.5%**	**93.8%**	**76.1%**	**73.7%**	**71.3%**	**77.1%**	**74.9%**	**85.0%**	↑ **8.9%**
CoT+GWO/n=6	**96.5%**	**81.5%**	**92.9%**	**99.8%**	**94.5%**	**76.8%**	**74.9%**	**72.2%**	**78.3%**	**75.8%**	**95.1%**	↑ **10.7%**
GPT-4.1-mini												
CoT	88.2%	79.8%	86.1%	99.0%	91.7%	67.5%	66.9%	69.5%	54.9%	71.2%	82.5%	-
CoT+GWO/n=5	**97.2%**	**83.6%**	**93.6%**	**99.8%**	**94.2%**	**79.3%**	**78.9%**	**77.2%**	**79.3%**	**78.8%**	**85.0%**	↑ **8.1%**
CoT+GWO/n=6	**98.3%**	**84.3%**	**94.8%**	**99.9%**	**94.8%**	**80.4%**	**79.8%**	**78.3%**	**80.5%**	**79.5%**	**97.4%**	↑ **10.1%**
GPT-4.1-nano												
CoT	83.8%	74.3%	81.1%	98.8%	89.6%	64.7%	61.3%	65.7%	52.1%	64.2%	80.0%	-
CoT+GWO/n=5	**93.1%**	**80.1%**	**92.3%**	**99.4%**	**93.1%**	**77.1%**	**74.1%**	**70.3%**	**77.8%**	**72.3%**	**83.0%**	↑ **8.8%**
CoT+GWO/n=6	**94.2%**	**81.2%**	**93.8%**	**99.7%**	**94.0%**	**78.2%**	**75.1%**	**71.2%**	**79.3%**	**73.5%**	**93.5%**	↑ **10.7%**
Qwen2.5-Coder-7B-Instruct												
CoT	77.5%	65.8%	82.7%	84.6%	87.3%	60.9%	55.4%	47.4%	31.3%	20.4%	70.4%	-
CoT+GWO/n=5	**89.7%**	**72.8%**	**90.6%**	**97.7%**	**90.5%**	**62.5%**	**58.5%**	**54.8%**	**37.7%**	**28.1%**	**75.5%**	↑ **6.8%**
CoT+GWO/n=6	**90.6%**	**73.8%**	**91.5%**	**98.1%**	**92.0%**	**63.1%**	**59.1%**	**55.6%**	**39.2%**	**26.1%**	**90.2%**	↑ **8.7%**
Gemma-3-12b-it												
CoT	83.5%	72.8%	79.3%	82.7%	91.0%	69.2%	68.4%	64.9%	78.0%	49.3%	75.1%	-
CoT+GWO/n=5	**93.4%**	**80.7%**	**91.3%**	**96.2%**	**92.1%**	**79.3%**	**73.1%**	**68.1%**	**85.3%**	**53.5%**	**81.5%**	↑ **7.3%**
CoT+GWO/n=6	**94.3%**	**81.3%**	**92.4%**	**96.8%**	**93.1%**	**80.5%**	**74.2%**	**68.8%**	**86.7%**	**54.4%**	**94.3%**	↑ **9.3%**

Table 2: Combined evaluation of adaptability of GWO to CoT on math and hybrid reasoning tasks.

leveraging dynamic multi-agent interactions to iteratively calibrate and consolidate the reasoning process, it mitigates error accumulation in long-chain inference and improves robustness and generalization in challenging reasoning scenarios. More details of the evaluation pipeline are provided in Appendix A.4.

4.3 Adaptability Evaluation

To further assess the adaptability and generalization of Agent-GWO, we integrate it with the mainstream reasoning paradigm CoT and evaluate the combined method across a broad set of mathematical and hybrid reasoning benchmarks. Specifically, we treat standard CoT prompting as the baseline and measure the accuracy gains obtained by incorporating Agent-GWO into the CoT pipeline. All runs are conducted independently, and accuracy is computed on the official validation sets of each benchmark. The aggregated results are reported in Table 2.

Across six math and five hybrid reasoning benchmarks, CoT+GWO yields consistent and monotonic gains over vanilla CoT for all evaluated backbones, with larger improvements as the iteration budget n increases. For GPT-4o-mini, CoT+GWO boosts GSM8K/MATH/AQUA from 85.4/74.8/65.3 to 95.1/79.9/76.1 with n=5, and further to 96.5/81.5/76.8 with n=6. When averaging over all 11 benchmarks, this corresponds to an absolute accuracy improvement of 8.9 percentage points for n=5 and 10.7 percentage points for n=6 relative to CoT. This consistent trend extends across various model families and scales, including Gemma and Qwen, as well as hybrid reasoning tasks. The gains are particularly pronounced on the MATH_MIX benchmark, where Qwen2.5-Coder-7B-Instruct achieved a substantial accuracy increase from 70.4% to 90.2%.

Overall, Table 2 shows that integrating Agent-GWO with CoT consistently improves reasoning performance across diverse benchmarks and model architectures, spanning mathematical, logical, temporal, and mixed-domain tasks. The consistent gains observed on both lightweight and stronger backbones highlight the generality and transferability of the proposed inference-time optimization framework, positioning Agent-GWO as an effective complement to standard CoT prompting for enhancing reasoning robustness and task alignment.

4.4 Hyper-parameter Sensitivity Analysis

To systematically evaluate the robustness and stability of Agent-GWO under decoding-level configurations, we conduct a sensitivity analysis on five key agent-level hyperparameters: temperature t_j, top-p threshold p_j, frequency penalty f_j, presence penalty e_j, and maximum generation length m_j. These hyperparameters regulate the stochasticity, diversity, repetition, novelty, and reasoning budget of LLM-based agents, and collectively shape agent behaviors during the optimization process. For continuous decoding parameters (t_j, p_j, f_j, e_j), each agent samples its configuration from a nor-

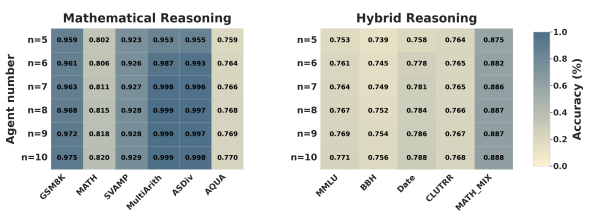

Figure 3: Accuracy over varying number of agents on mathematical (right) and hybrid (left) reasoning tasks.

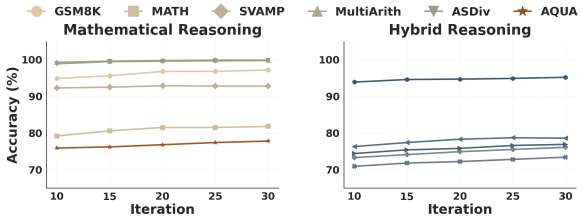

Figure 4: Accuracy over iteration for math (left) and hybrid (right) reasoning tasks (fixed five agents).

Parameter	Sampling Rule	Perf.(%)	Cost ($)
t_j	(0.2, 0.1)	↓1.0–1.5%	↓3–5%
	(0.4, 0.1)	↓0.3–0.6%	↓1–2%
	(0.6, 0.1)	baseline	baseline
	(0.8, 0.1)	↓0.5–1.0%	↑4–7%
p_j	(0.2, 0.1)	↓0.8–1.3%	↓2–4%
	(0.4, 0.1)	↓0.2–0.5%	↓1–2%
	(0.6, 0.1)	baseline	baseline
	(0.8, 0.1)	↑0–0.3%	↑2–4%
f_j	(0.0, 0.1)	↓0.6–1.0%	↑4–6%
	(0.4, 0.1)	↓0.2–0.4%	↑1–2%
	(0.6, 0.1)	baseline	baseline
	(0.8, 0.1)	±0–1%	↓1–3%
e_j	(0.0, 0.1)	↓0.4–0.8%	↑2–4%
	(0.4, 0.1)	↓0.1–0.3%	↑1–2%
	(0.6, 0.1)	baseline	baseline
	(0.8, 0.1)	↓0.5–1.0%	↑3–6%
m_j	[1024,1274]	↓0.5–1.5%	↓5–12%
	[1274,1524]	baseline	baseline
	[1524,1774]	↑0–0.4%	↑3–8%
	[1774,2024]	↑0–0.8%	↑6–15%

Table 3: Sensitivity analysis of agent-level decoding hyperparameters. Entries report relative performance and cost changes compared to the baseline configuration used in our framework.

mal distribution $\mathcal{N}(\mu, \sigma^2)$, followed by clipping to a valid range. This strategy introduces controlled variability across agents while preventing unstable or degenerate decoding settings. The maximum generation length m_j is treated separately and sampled uniformly from predefined intervals, enabling explicit control over the available reasoning budget and inference cost.

Table 3 reports the relative changes in task performance and inference cost under different decoding hyperparameter settings, normalized to the default configuration. All sensitivity experiments are conducted on the gpt-4o-mini model using the GSM8K dataset. Empirical results show that Agent-GWO is largely insensitive to moderate variations in decoding hyperparameters. Across a wide range of (μ, σ) settings and token-length intervals, performance fluctuations remain limited, typically within $\pm 1.5\%$. Moderate values of t_j and p_j consistently yield stable performance, while extreme settings tend to degrade accuracy due to insufficient diversity or excessive randomness. The penalty terms f_j and e_j mainly affect output redundancy and repetition, with limited influence on accuracy but observable impact on inference cost. Among all parameters, m_j exhibits the most direct trade-off between performance and computational cost.

Overall, this analysis suggests that Agent-GWO does not rely on delicate hyperparameter tuning to achieve stable performance, supporting its practical reliability and reproducibility across different decoding configurations.

4.5 Ablation Study

Impact of Agent Population and Iteration. To analyze the contribution of the number of agents and iterations in the GWO framework, we conduct an ablation study on GPT-4o-mini across multiple datasets. We first examine the effect of varying the number of agents, with $n \in \{5, 6, 7, 8, 9, 10\}$. As shown in Figure 3, accuracy on both mathematical (GSM8K, MATH, SVAMP, MultiArith, ASDiv, AQUA) and hybrid reasoning benchmarks (MMLU, BBH, Date, CLUTRR, MATH_MIX) generally improves as the number of agents increases. For example, GSM8K accuracy rises from 95.9% at $n = 5$ to 97.5% at $n = 10$, while MMLU improves from 75.3% to 77.1%. The gains, however, are non-linear and diminish as n grows, indicating that moderate increases in agent count yield limited additional benefits and should be weighed against computational cost.

We further study the effect of the number of optimization iterations while fixing the number of agents at $n = 5$. As illustrated in Figure 4, increasing the number of iterations from 10 to 30 consistently improves performance across both mathematical and hybrid reasoning tasks. This trend suggests that additional iterations enable more thor-

Elite Size	Elite	GSM8K	MATH	MMLU
$m = 1$	α	92.4	76.8	72.0
$m = 2$	α, β	94.8	79.1	74.2
$m = 3$	α, β, δ	**95.9**	**80.2**	**75.3**
$m = 4$	$\alpha, \beta, \delta, \omega_1$	95.0	79.4	74.6
$m = 5$	$\alpha, \beta, \delta, \omega_1, \omega_2$	94.1	78.5	73.8

Table 4: Ablation on Elite Set Size. Results under different elite set sizes m. α, β, and δ denote the top three leaders in standard GWO, while ω_1 and ω_2 denote the 4th- and 5th-ranked agents by fitness, respectively.

ough exploration of the solution space and progressive refinement of reasoning strategies, leading to more stable convergence. At the dataset level, most benchmarks exhibit monotonic or near-monotonic improvements as the iteration count increases, underscoring the importance of iterative optimization in GWO.

Impact of the Number of Elite Agents. To understand how the diversity of the guidance signal affects the search process, we investigate the sensitivity of our method to the number of elite agents, denoted as m. In standard GWO, the search is guided by the top-three leaders (α, β, and δ). To verify if this configuration is optimal for our discrete search space, we vary m from 1 to 5. As illustrated in Table 4, we observe a clear inverted U-shaped trend across the GSM8K, MATH, and MMLU benchmarks. Increasing m from 1 to 3 steadily improves performance, as the added leader diversity helps the algorithm avoid premature convergence. However, when $m > 3$, the inclusion of lower-ranked agents (e.g., ω_1 and ω_2) mixes suboptimal prompts into the update step. This dilutes the strong guidance signal from the top leaders and reduces the overall selection pressure, leading to a slight drop in accuracy. Consequently, we maintain $m = 3$ as the default configuration.

Overall, the number of agents, iterations, and elite set size all affect the effectiveness of GWO. Increasing the agent population enhances global exploration, while additional iterations improve convergence stability and reduce variance, and a proper elite set size helps balance guidance diversity and selection pressure. Together, these factors support the scalability of the GWO framework, enabling it to better leverage the reasoning potential of large models across various tasks.

5 Conclusion

In this paper, we propose Agent-GWO, a dynamic prompt optimization framework tailored for complex reasoning tasks. By unifying prompt templates and decoding configurations into iteratively optimizable agent states, we design a hierarchical collaborative mechanism that enables agents to progressively refine strategies through mutual learning over multiple iterations. This swarm-based perspective facilitates stable convergence while effectively maintaining exploration capabilities, allowing for the automatic discovery of robust, task-aligned reasoning configurations that are directly deployable for inference. Extensive experiments across diverse reasoning benchmarks and varying computational budgets demonstrate that Agent-GWO consistently outperforms both static prompting baselines and existing automatic prompt optimization methods in terms of accuracy and stability.

Limitations

Agent-GWO relies on parallel exploration with multiple agents and iterative updates at test time; accordingly, it incurs a higher inference-time compute budget than single-pass prompting and should be configured with appropriate (n, K) under practical compute and latency constraints. In addition, our empirical study primarily focuses on mathematical and hybrid reasoning benchmarks. While the proposed framework is broadly applicable in form, further evaluations on a wider range of domain-specific and real-world tasks are needed to more fully characterize its effectiveness and operating boundaries.

In future work, we will further improve the efficiency and generalization of the framework. One important direction is to develop more compute-aware agent scheduling and adaptive update strategies to reduce the cost of multi-agent exploration while maintaining optimization quality. Another direction is to extend Agent-GWO to broader application scenarios, including domain-specific reasoning tasks and open-ended real-world tasks, so as to more comprehensively evaluate its robustness and transferability.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China under grant 62572104, and 62220106008.

References

Nurul Fajrin Ariyani, Zied Bouraoui, Richard Booth, and Steven Schockaert. 2025. There's no such thing as simple reasoning for llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4503–4514.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. *Advances in Neural Information Processing Systems*, 37:69022–69042.

Sihan Cao, Jianwei Zhang, Pengcheng Zheng, Jiaxin Yan, Caiyan Qin, Yalan Ye, Wei Dong, Peng Wang, Yang Yang, and Chaoning Zhang. 2026. Language-guided token compression with reinforcement learning in large vision-language models. *arXiv preprint arXiv:2603.13394*.

Anwoy Chatterjee, HSVNS Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. 2024. Posix: A prompt sensitivity index for large language models. *arXiv preprint arXiv:2410.02185*.

Xinyun Chen, Jacob Tworek, Heewoo Jun, Qiming Yuan, Maarten Bosma, Yi Luan, Denny Zhou, Quoc V Le, and Luke Zettlemoyer. 2022. Program-of-thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. In *Advances in Neural Information Processing Systems*, volume 35, pages 24842–24857.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Jacques Ferber and Gerhard Weiss. 1999. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-wesley Reading.

Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698.

ACL Fipa. 2002. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004)*.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zy Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and 1 others. 2025. Can large language models detect errors in long chain-of-thought reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18468–18489.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, and 1 others. 2025. Data interpreter: An llm agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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

Fanding Huang, Guanbo Huang, Xiao Fan, Yi He, Xiao Liang, Xiao Chen, Qinting Jiang, Faisal Nadeem Khan, Jingyan Jiang, and Zhi Wang. 2026. Semantic-space exploration and exploitation in rlvr for llm reasoning. *Preprint*, arXiv:2509.23808.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42.

Jiechuan Jiang and Zongqing Lu. 2018. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Pranjal Kumar. 2024. Large language models (llms): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260.

Joshua Ong Jun Leang, Aryo Pradipta Gema, and Shay B Cohen. 2025. Comat: Chain of mathematically annotated thought improves mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20256–20285.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Chunyang Li, Weiqi Wang, Tianshi Zheng, and Yangqiu Song. 2025a. Patterns over principles: The fragility of inductive reasoning in llms under noisy observations. *arXiv preprint arXiv:2502.16169*.

Junlong Li, Daya Guo, Dejian Yang, Runxin Xu, Yu Wu, and Junxian He. 2025b. Codei/o: Condensing reasoning patterns via code input-output prediction. *arXiv preprint arXiv:2502.07316*.

Nian Li, Chen Gao, Mingyu Li, Yong Li, and Qingmin Liao. 2023. Econagent: large language model-empowered agents for simulating macroeconomic activities. *arXiv preprint arXiv:2310.10436*.

Jonghan Lim, Birgit Vogel-Heuser, and Ilya Kovalenko. 2024. Large language model-enabled multi-agent manufacturing systems. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 3940–3946. IEEE.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. https://github.com/google-deepmind/AQuA. NeurIPS Datasets Track Reference.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, and 1 others. 2025. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*.

Yujun Liu, Xinyun Chen, Jacob Tworek, Qiming Yuan, Maarten Bosma, Yi Luan, Denny Zhou, Quoc V Le, and Luke Zettlemoyer. 2023. Graph-of-thought prompting: Structuring reasoning as dependency graphs. In *Advances in Neural Information Processing Systems*, volume 36, pages 10243–10257.

Xinyun Lu, Xiujun Li, Di He, Jianfeng Gao, and Li Deng. 2022. Fantastically ordered prompts and where to find them. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 8000–8011.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Mohammad Reza Ghasemi Madani, Soyeon Caren Han, Shuo Yang, and Jey Han Lau. 2026. Inclusion-of-thoughts: Mitigating preference instability via purifying the decision space. *arXiv preprint arXiv:2604.04944*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pages 975–984.

Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. 2014. Grey wolf optimizer. *Advances in engineering software*, 69:46–61.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2080–2094.

Ofir Press, Mike Lewis Lee, and Luke Zettlemoyer. 2022. Self-ask with search: Bootstrapping reasoning with chain-of-thought. In *Advances in Neural Information Processing Systems*, volume 35, pages 24804–24817.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1743–1752.

Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Guha, E Kelly Buchanan, Mayee Chen, Neel Guha, Christopher Ré, and 1 others. 2024. Archon: An architecture search framework for inference-time techniques. *arXiv preprint arXiv:2409.15254*.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515.

Saksham Sahai Srivastava and Ashutosh Gandhi. 2024. Mathdivide: Improved mathematical reasoning by large language models. *arXiv preprint arXiv:2405.13004*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. 2025. Atom of thoughts for markov llm test-time scaling. *arXiv preprint arXiv:2502.12018*.

Chengbing Wang, Yang Zhang, Wenjie Wang, Xiaoyan Zhao, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2025. Think-while-generating: On-the-fly reasoning for personalized long-form generation. *arXiv preprint arXiv:2512.06690*.

Junxi Wang, Te Sun, Jiayi Zhu, Junxian Li, Haowen Xu, Zichen Wen, Xuming Hu, Zhiyu Li, and Linfeng Zhang. 2026. Streammeco: Long-term agent memory compression for efficient streaming video understanding. *Preprint*, arXiv:2604.09000.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Shijia Xu, Yu Wang, Xiaolong Jia, Zhou Wu, Kai Liu, and April Xiaowen Dong. 2026. Rcbsf: A multi-agent framework for automated contract revision via stackelberg game. *Preprint*, arXiv:2604.10740.

Shuo Yang, Soyeon Caren Han, Yihao Ding, Shuhe Wang, and Eduard Hoy. 2026a. Tooltree: Efficient llm agent tool planning via dual-feedback monte carlo tree search and bidirectional pruning. *arXiv preprint arXiv:2603.12740*.

Shuo Yang, Soyeon Caren Han, Xueqi Ma, Yan Li, Mohammad Reza Ghasemi Madani, and Eduard Hovy. 2026b. Evotool: Self-evolving tool-use policy optimization in llm agents via blame-aware mutation and diversity-aware selection. *arXiv preprint arXiv:2603.04900*.

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

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2023. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*.

An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024a. On generative agents in recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in Information Retrieval*, pages 1807–1817.

Jiaquan Zhang, Qigan Sun, Chaoning Zhang, Xudong Wang, Zhenzhen Huang, Yitian Zhou, Pengcheng Zheng, Chi-lok Andy Tai, Sung-Ho Bae, Zeyu Ma, and 1 others. 2026a. Tda-rc: Task-driven alignment for knowledge-based reasoning chains in large language models. *arXiv preprint arXiv:2604.04942*.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Zhenzhen Huang, Pengcheng Zheng, Zhicheng Wang, Ping Guo, Fan Mo, Sung-Ho Bae, Jie Zou, Jiwei Wei, and Yang Yang. 2026b. Lightweight llm agent memory with small language models. *Preprint*, arXiv:2604.07798.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Yibei Liu, Chenghao Li, Qigan Sun, Shuai Yuan, Fachrina Dewi Puspitasari, Dongshen Han, Guoqing Wang, and 1 others. 2026c. Text summarization via global structure awareness. *arXiv preprint arXiv:2602.09821*.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Xudong Wang, Zhenzhen Huang, Pengcheng Zheng, Shuai Yuan, Sheng Zheng, Qigan Sun, Jie Zou, and 1 others. 2026d. Learning global hypothesis space for enhancing synergistic reasoning chain. *arXiv preprint arXiv:2602.09794*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2024b. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024c. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Pengcheng Zheng, Xiaorong Pu, Kecheng Chen, Jiaxin Huang, Meng Yang, Bai Feng, Yazhou Ren, Jianan Jiang, Chaoning Zhang, Yang Yang, and 1 others. 2025. Joint lossless compression and steganography for medical images via large language models. *arXiv preprint arXiv:2508.01782*.

Pengcheng Zheng, Chaoning Zhang, Mingzheng Cui, Guo Chen, Qigan Sun, Jiaxin Huang, Jiaquan Zhang, Tae-Ho Kim, Caiyan Qin, Yazhou Ren, and 1 others. 2026a. Towards visual chain-of-thought reasoning: A comprehensive survey.

Pengcheng Zheng, Chaoning Zhang, Jiarong Mo, Guo-Hui Li, Jiaquan Zhang, Jiahao Zhang, Sihan Cao, Sheng Zheng, Caiyan Qin, Guoqing Wang, and 1 others. 2026b. Llava-fa: Learning fourier approximation for compressing large multimodal models. *arXiv preprint arXiv:2602.00135*.

Denny Zhou, Dale Schuurmans, Quoc V Le, Ed H Chi, and Jason Wei. 2022. Least-to-most prompting enables complex reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24770–24782.

A Appendix

A.1 Preliminary Knowledge for Grey Wolf Optimizer

Grey Wolf Optimizer. GWO is a population-based metaheuristic inspired by the social hierarchy and hunting behavior of grey wolves. Designed for solving continuous optimization problems, GWO is characterized by its simplicity and minimal reliance on hyperparameters. In this algorithm, a population of N wolves is maintained, where each wolf represents a candidate solution $\mathbf{X}_i = (x_i^1, x_i^2, \ldots, x_i^D)$ in a D-dimensional search space. The objective is to optimize a target function $f(\mathbf{X})$. GWO mimics the hierarchical structure of grey wolves, dividing the population into four roles: α (leader), β (second-in-command), δ (subordinate), and ω (follower). During the optimization process, the α, β, and δ wolves correspond to the top three solutions and are responsible for guiding the remaining ω wolves toward promising regions in the search space, as illustrated in Figure 5.

The core mechanism of the GWO is inspired by the encircling behavior exhibited during hunting. In each iteration, wolves estimate their distance from the prey (i.e., the current best solution) as $\mathbf{D}(t) = |\mathbf{C}(t) \odot \mathbf{X}_p(t) - \mathbf{X}_i(t)|$, where $\mathbf{X}_p(t)$ denotes the position of the prey at iteration t, $\mathbf{C}(t) = 2\mathbf{r}_1(t)$, and $\mathbf{r}_1(t) \in [0,1]^D$ is a uniformly distributed random vector.

Based on $\mathbf{D}(t)$, the position of each wolf is updated by $\mathbf{X}_i(t+1) = \mathbf{X}_p(t) - \mathbf{A}(t) \odot \mathbf{D}(t)$, where $\mathbf{A}(t) = 2\,\mathbf{a}(t) \odot \mathbf{r}_2(t) - \mathbf{a}(t)$ and $\mathbf{r}_2(t) \in [0,1]^D$ is another random vector. The parameter decreases linearly with iterations as $\mathbf{a}(t) = 2 - 2 \cdot \frac{t}{T_{\max}}$, where $T_{\max}$ is the maximum number of iterations. This design allows the algorithm to balance exploration in early stages and exploitation in later stages.

Each ω wolf also updates its position by referring to the three leading wolves α, β, and δ. The distances are computed as $\mathbf{D}_j(t) = |\mathbf{C}_j(t) \odot \mathbf{X}_j(t) - \mathbf{X}_i(t)|$, $j \in \{\alpha, \beta, \delta\}$, and the corresponding candidate updates are $\mathbf{X}_i^{(j)}(t+1) = \mathbf{X}_j(t) - \mathbf{A}_j(t) \odot \mathbf{D}_j(t)$. Finally, the new position of wolf i is obtained by averaging the three guided positions:

$$\mathbf{X}_i(t+1) = \frac{\mathbf{X}_i^{(\alpha)}(t+1) + \mathbf{X}_i^{(\beta)}(t+1) + \mathbf{X}_i^{(\delta)}(t+1)}{3} \quad (5)$$

By integrating leader-based guidance with stochastic exploration, the GWO effectively

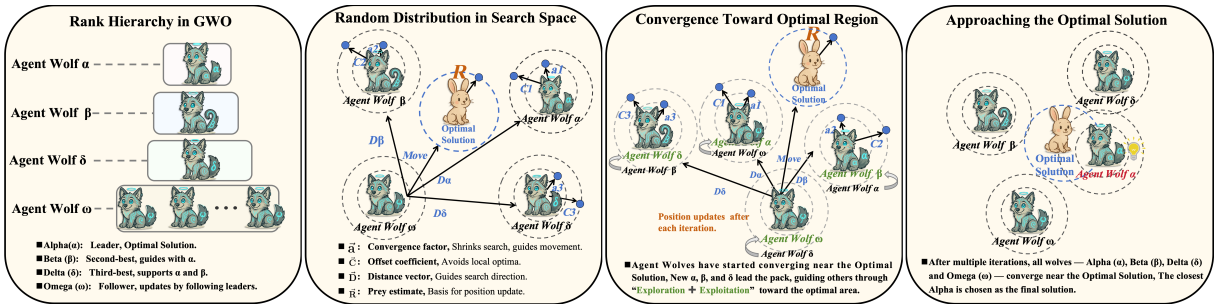

Figure 5: Illustration of GWO algorithm.

achieves a balance between global search and local exploitation. As iterations progress, the decreasing parameter $\mathbf{a}(t)$ ensures a smooth transition from exploration to exploitation, enhancing convergence toward the global optimum.

A.2 LLM-as-a-Judge Protocol

We adopt a controlled *LLM-as-a-Judge* protocol as an auxiliary evaluation mechanism to assess the quality of intermediate reasoning states and state transitions, particularly in settings where explicit gold answers are unavailable.

Evaluation Setup. A fixed large language model is used as the judge across all experiments. The judge model, prompt, and decoding configuration are kept identical throughout training and evaluation. The prompt is frozen prior to experimentation and is not adapted based on agent outputs or historical performance, preventing self-referential evaluation.

For each transition, the judge is provided with the current state Q_i, the candidate next state Q_{i+1}, and their corresponding reasoning trajectories. The judge evaluates whether the transition preserves the original reasoning objective while offering improved reasoning utility.

Evaluation Criteria. The judge produces three normalized scores in $[0, 1]$:

$$\left(s^{\text{logic}},\ s^{\text{creativity}},\ s^{\text{complete}}\right),$$

corresponding to logical consistency, structural novelty introduced by the transition, and coverage of required reasoning elements, respectively. The judge is explicitly instructed not to rely on superficial cues such as verbosity or stylistic features.

Usage in Different Settings. For verifiable benchmarks with gold answers, exact-match accuracy remains the primary fitness signal. Judge-based scores are used only as tie-breakers when multiple agents achieve identical accuracy. For non-verifiable tasks, agent fitness is computed as a weighted sum of judge scores:

$$\text{FITNESS}_j = w_1 s^{\text{logic}} + w_2 s^{\text{creativity}} + w_3 s^{\text{complete}},$$

with weights $(w_1, w_2, w_3) = (0.5, 0.2, 0.3)$ calibrated on 200 human-labeled samples.

Stability. All judge-based scores are averaged over three independent runs with different random seeds to reduce variance and improve ranking stability.

A.3 Pseudocode

Algorithm 1 Grey Wolf Optimizer (GWO)

Require: Population size N, dimension D, objective $\mathcal{F}$, max iterations $T_{\max}$
Ensure: Best solution $\mathbf{X}^\star$
Initialization
Initialize population $\{\mathbf{X}_i\}_{i=1}^{N} \subset \mathbb{R}^D$ within bounds
$a \leftarrow 2$
$\mathbf{X}^\star \leftarrow \arg\min_{\mathbf{X}_i}\ \mathcal{F}(\mathbf{X}_i)$
Iterative Optimization
for $t = 1$ **to** $T_{\max}$ **do**
  Evaluate $\mathcal{F}(\mathbf{X}_i)$ for all i
  Identify leader wolves α, β, δ (top-3 by fitness)
  $a \leftarrow 2 - 2t/T_{\max}$
  for all $\mathbf{X}_i$ **do**
    for all $\ell \in \{\alpha, \beta, \delta\}$ **do**
      Sample $\mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{U}(0,1)^D$
      $\mathbf{A}_\ell \leftarrow 2a\mathbf{r}_1 - a$
      $\mathbf{C}_\ell \leftarrow 2\mathbf{r}_2$
      $\mathbf{D}_\ell \leftarrow |\mathbf{C}_\ell \odot \mathbf{X}_\ell - \mathbf{X}_i|$
      $\mathbf{X}'_\ell \leftarrow \mathbf{X}_\ell - \mathbf{A}_\ell \odot \mathbf{D}_\ell$
    end for
    $\mathbf{X}_i \leftarrow \frac{1}{3}(\mathbf{X}'_\alpha + \mathbf{X}'_\beta + \mathbf{X}'_\delta)$
    Project $\mathbf{X}_i$ to feasible bounds
  end for
  Update $\mathbf{X}^\star$ if improved
end for
return $\mathbf{X}^\star$

Algorithm 1 presents the standard GWO: evaluate all candidates, select the top three leaders (α, β, δ), update the rest by aggregating leader-guided proposals with annealed coefficients, project to bounds, and track the best $\mathbf{X}^\star$.

Algorithm 2 Agent-Level Iterative Optimization via GWO

Require: Dataset $\mathcal{D}$, number of agents n, iterations K
Ensure: Optimal agent configuration $(\boldsymbol{\eta}^*, prompt^*)$

Initialization

Initialize empty agent set $\mathcal{A} \leftarrow \emptyset$
for $j = 1$ **to** n **do**
  Sample hyperparameters $\boldsymbol{\eta}_j \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
  $\boldsymbol{\eta}_j \leftarrow \mathrm{Clip}(\boldsymbol{\eta}_j)$
  Sample prompt template $prompt_j$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{(\boldsymbol{\eta}_j, prompt_j)\}$
end for

Iterative Optimization

for $k = 1$ **to** K **do**
  for all $(\boldsymbol{\eta}_j, prompt_j) \in \mathcal{A}$ **do**
    Sample question $q \sim \mathcal{D}$
    $(CoT_j, Ans_j) \leftarrow \mathrm{LLM}(\boldsymbol{\eta}_j, prompt_j, q)$
    $Fitness_j \leftarrow \mathrm{Evaluate}(Ans_j, q)$
  end for
  Select leader agents (α, β, δ) by fitness
  Define sampling weights $\mathbf{w} = (w_\alpha, w_\beta, w_\delta)$, $\sum w = 1$
  for all $(\boldsymbol{\eta}_j, prompt_j) \in \mathcal{A} \setminus \{\alpha, \beta, \delta\}$ **do**
    Sample leader $r \sim \{\alpha, \beta, \delta\}$ according to $\mathbf{w}$
    $\boldsymbol{\eta}_j \leftarrow \mathrm{Clip}(\mathcal{N}(\boldsymbol{\eta}_r, \sigma^2))$
    $prompt_j \leftarrow \mathrm{PromptAdaptation}(prompt_j, \{prompt_\alpha, prompt_\beta, prompt_\delta\})$
  end for
end for

Output

Extract best agent $(\boldsymbol{\eta}^*, prompt^*) \leftarrow (\boldsymbol{\eta}_\alpha, prompt_\alpha)$
return $(\boldsymbol{\eta}^*, prompt^*)$

Algorithm 2 instantiates GWO for our setting, where each agent is $(\boldsymbol{\eta}_j, prompt_j)$. Each iteration scores agents on sampled questions, selects leaders, and updates non-leaders by following leader hyperparameters (perturb + clip) and adapting prompts via $\mathrm{PromptAdaptation}(\cdot)$, returning the best leader $(\boldsymbol{\eta}^*, prompt^*)$.

A.4 More Experimental Results

To further evaluate the effectiveness of our GWO-based prompt optimization, we conduct benchmarks on two standard math reasoning datasets, GSM8K and MATH, using Qwen2.5-Coder-7B-Instruct as the primary backbone. Table 6 reports accuracy (%) for representative prompt optimization and reasoning baselines. Compared to vanilla CoT and CoT with self-consistency sampling (CoT-SC), GWO yields a substantial improvement, reaching 89.1% on GSM8K and 72.1% on MATH. When combined with CoT, the performance is further strengthened, and our best setting (GWO/n=6 + CoT) achieves 90.6% on GSM8K and 73.8% on MATH, outperforming the listed strong baselines under the same backbone.

In addition, we compare against widely-used GPT-4 prompting/optimization baselines on GSM8K to contextualize the strength of the resulting configuration. As shown in Table 5, GWO/n=6 + CoT with GPT-4o-mini attains 96.5% accuracy on GSM8K, which is competitive with multiple optimized prompting methods reported for GPT-4. Overall, these results indicate that GWO consistently improves upon CoT-style prompting across backbones and datasets, and its combination with CoT is complementary, providing a simple yet effective way to obtain stronger and more stable reasoning performance.

Table 5: Accuracy on GSM8K comparing GPT-4 and GPT-4o-mini.

Method	Model	GSM8K
CoMAT (Leang et al., 2025)	GPT-4	93.7
CoT (Wei et al., 2022)	GPT-4	94.5
FCoT (Lyu et al., 2023)	GPT-4	95.0
MathPrompter (Imani et al., 2023)	GPT-4	95.6
QuaSAR (Radford et al., 2018)	GPT-4	96.5
MathDivide (Srivastava and Gandhi, 2024)	GPT-4	96.8
GWO/n=6+CoT	**GPT-4o-mini**	**96.5**

Table 6: Accuracy (%) on GSM8K and MATH using Qwen2.5-Coder-7B.

Method	Base	GSM8K	MATH
OMI2 (Li et al., 2025b)	Qwen2.5	84.1	72.3
CODEI/O++ (Li et al., 2025b)	Qwen2.5	85.7	72.1
PyEdu (Li et al., 2025b)	Qwen2.5	85.8	71.4
CODEI/O (Li et al., 2025b)	Qwen2.5	86.4	71.9
OC-SFT-1 (Li et al., 2025b)	Qwen2.5	86.7	70.9
WI (Li et al., 2025b)	Qwen2.5	87.0	71.4
WI (Full) (Li et al., 2025b)	Qwen2.5	87.0	71.1
OMI2 (Full) (Li et al., 2025b)	Qwen2.5	88.5	73.2
CoT	Qwen2.5	77.3	69.7
CoT-SC/n=5	Qwen2.5	80.1	71.2
GWO	Qwen2.5	89.1	72.1
GWO+CoT	Qwen2.5	89.7	72.8
GWO/n=6+CoT	**Qwen2.5**	**90.6**	**73.8**

A.5 Notation

Symbol	Definition
Grey Wolf Optimizer (GWO) Symbols	
N	Population size (number of wolves/solutions)
D	Search space dimensionality
$\mathbf{X}_i$	Position of the i-th wolf: $\mathbf{X}_i = (x_i^1, x_i^2, \ldots, x_i^D)$
$f(\mathbf{X})$	Objective function to be optimized
$T_{\max}$	Maximum number of iterations
t	Current iteration number
$\mathbf{X}_p(t)$	Prey position at iteration t (current best solution)
$\mathbf{C}$	Coefficient vector, $\mathbf{C} = 2\mathbf{r}_1$, $\mathbf{r}_1 \sim \mathcal{U}(0,1)^D$
$\mathbf{D}$	Distance vector, $\mathbf{D} = \lvert\mathbf{C} \cdot \mathbf{X}_p(t) - \mathbf{X}_i(t)\rvert$
$\mathbf{A}$	Adaptive coefficient, $\mathbf{A} = 2\mathbf{a} \cdot \mathbf{r}_2 - \mathbf{a}$, $\mathbf{r}_2 \sim \mathcal{U}(0,1)^D$
$\mathbf{a}$	Linearly decreasing parameter, $\mathbf{a} = 2 - 2 \cdot \frac{t}{T_{\max}}$
α, β, δ	The top three wolves (leaders) in GWO
ω	The remaining wolves (followers)
$\mathbf{D}_j$	Distance from wolf i to leader j: $\mathbf{D}_j = \lvert\mathbf{C}_j \cdot \mathbf{X}_j - \mathbf{X}_i\rvert$, $j \in \{\alpha, \beta, \delta\}$
$\mathbf{A}_j, \mathbf{C}_j$	Adaptive/random coefficients for leader j
$\mathbf{X}'_j$	Updated position guided by leader j
$\mathcal{F}$	Fitness function
Agent Symbols	
n	Number of agents
Agent_j	The j-th agent, $\text{Agent}_j = \{\text{LLM}_j, \text{prompt}_j\}$
LLM_j	Large language model of agent j, $\text{LLM}_j = \{\boldsymbol{\eta}_j, \boldsymbol{\theta}\}$
$\boldsymbol{\theta}$	Shared model parameter set of LLMs
$\boldsymbol{\eta}_j$	Agent-specific hyperparameter set: $\{T_j, p_j, F_j, E_j, M_j\}$
T_j	Temperature hyperparameter for agent j
p_j	Top-p threshold for agent j
F_j	Frequency penalty for agent j
E_j	Presence penalty for agent j
M_j	Maximum token length for agent j
$\text{clip}(x, [a, b])$	Clipping function: $\max(a, \min(x, b))$
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
$\mathcal{U}(a, b)$	Uniform distribution over $[a, b]$
$\mathcal{M}$	Discrete set of possible maximum token lengths
c_M	Constant value for maximum token length (fixed-length tasks)
Data and Task Symbols	
$\mathcal{D}$	Reasoning problem dataset, $\mathcal{D} = \{q_1, q_2, \ldots, q_N\}$
q	A single question from dataset $\mathcal{D}$
CoT_j	Chain of Thought generated by agent j
Answer_j	Final answer generated by agent j
$f(\text{Agent}_j, q)$	Output of agent j on q: $(\text{CoT}_j, \text{Answer}_j)$
Optimization Process Symbols	
K	Number of optimization iterations in multi-agent GWO
$\mathcal{A}$	Population set of agents
$\mathbf{w}$	Weight vector for leaders: $\mathbf{w} = \{w_\alpha, w_\beta, w_\delta\}$, $w_\alpha > w_\beta > w_\delta$, $\sum w = 1$
promptAdaptation	prompt template adaptation function
$(\boldsymbol{\eta}^*, \textit{prompt}^*)$	Optimal hyperparameters and prompt template found

A.6 Training Examples

A.6.1 Forward Inference Examples

This example presents a math word problem about recycling cans. The problem describes that Tiffany was collecting cans for recycling. On Monday, she had 3 bags of cans, and the next day she found 7 more bags of cans and 44 bags of bottles. The question asks how many bags of cans she had altogether. The answer given in the image is 10 bags. By adding the number of bags of cans found on both days, we can conclude that Tiffany had a total of 10 bags of cans. This problem tests the ability to understand information from the question and perform simple addition.

Dataloading for Interation 1

Problem: YYYY-MM-DD 22:35:34,676 - log.logger_utils - INFO

Tiffany was collecting cans for recycling. On Monday, she had 3 bags of cans. The next day, she found 7 more bags of cans and 44 bags of bottles. How many bags of cans did she have altogether?

Answer: YYYY-MM-DD 22:35:34,676 - log.logger_utils - INFO

10

This example demonstrates the parameter settings, systematic problem-solving approach, and the beginning of the answer provided by the first agent when tackling a mathematical word problem. The focus is on the standardized process of problem-solving, emphasizing a complete workflow from understanding the question to checking the final answer.

Training for Agent 1 in Interaction 1

Prameter Log for Agent 1: YYYY-MM-DD 22:35:34,691 - log.logger_utils - INFO

temperature: 0.9193923355128871, top_p: 0.9027935433165666, frequency_penalty: 1.8091973063219, presence_penalty: -1.7443228132190376, max_tokens: 1919

prompt Log for Agent 1: YYYY-MM-DD 22:35:48,332 - log.logger_utils - INFO

To solve the questions found in the svamp dataset, follow these steps:
1. Carefully read the entire question to understand what is being asked.
2. Identify and extract all relevant numerical data and quantities mentioned within the question.
3. Determine which mathematical operations (addition, subtraction, multiplication, division) are necessary based on keywords or phrases that indicate relationships between numbers (e.g., "altogether," "more than," etc.).
4. Perform calculations step-by-step while keeping track of intermediate results if needed to avoid errors.
5. Double-check your final answer against the context of the problem to ensure it makes sense.
Use this approach systematically for each question from the SVAMP dataset you encounter!

Answer by Agent 1: YYYY-MM-DD 22:35:52,418 - log.logger_utils - INFO

Let's follow the steps to solve the problem:
1. Read the entire question: Tiffany collected cans for recycling, and we need to find out how many bags of cans she had altogether after her collection over two days.
2. Extract relevant numerical data:
On Monday, Tiffany had 3 bags of cans.
On Tuesday, she found 7 more bags of cans.
3. Determine necessary mathematical operations: We need to find the total number of bags of cans, which involves addition since we are combining quantities.
4. Perform calculations step-by-step:
Start with the number of bags she had on Monday: 3
Add the number of additional bags found on Tuesday:

$$\text{Total Bags} = 3 + 7 = 10$$

5. Double-check final answer against context: The question asks for how many bags of "cans" she has altogether, and we calculated that Tiffany has a total of 10 bags (only counting "cans").

Therefore, Tiffany has a total of 10 bags of cans altogether.

This example provides a detailed demonstration of how the first agent systematically solves a mathematical word problem using a five-step approach (understanding the problem, extracting data, determining the operation, step-by-step calculation, and checking the answer), ultimately arriving at a clear and accurate solution.

Training for Agent 2 in Interaction 1

Prameter Log for Agent 2: YYYY-MM-DD 22:35:52,532 - log.logger_utils - INFO
temperature: 0.39241099147650554, top_p: 0.6602020470317261, frequency_penalty: 0.0007142275093004872, presence_penalty: -0.19136457572912707, max_tokens: 1413

prompt Log for Agent 2: YYYY-MM-DD 22:36:05,253 - log.logger_utils - INFO
Create a word problem that involves basic arithmetic operations based on the given scenario. The problem should include specific quantities and require calculations to determine totals, differences, or other values relevant to everyday activities, ensuring that it can engage learners in applying their math skills within relatable contexts like shopping, collecting items, or planning events.

Answer by Agent 2: YYYY-MM-DD 22:36:09,262 - log.logger_utils - INFO
To find out how many bags of cans Tiffany had altogether, we need to focus only on the bags of cans.
1. On Monday, Tiffany had 3 bags of cans.
2. The next day, she found 7 more bags of cans.
Now, we can calculate the total number of bags of cans:
Total bags of cans = Bags on Monday + Bags found on Tuesday
Total bags of cans = 3 + 7
Now, let's do the calculation:
Total bags of cans = 3 + 7 = 10
So, Tiffany had 10 bags of cans altogether.
(Note: The bags of bottles are not included in the total for bags of cans, as the problem specifically asks for the total number of bags of cans.)

This example demonstrates how the second agent generates and solves a simple math word problem following a standardized process. It covers all key steps, including problem design, information extraction, mathematical operations, step-by-step calculation, and explanation of the result. The example also specifically clarifies the scope of the problem (counting only bags of cans, not bottles), which helps the agent understand the problem requirements and the solution approach.

Training for Agent 3 in Interation 1

Prameter Log for Agent 3: YYYY-MM-DD 22:36:09,369 - log.logger_utils - INFO
temperature: 0.5822851007143794, top_p: 0.9719972687061527, frequency_penalty: 0.04468780736250461, presence_penalty: 0.02409206816099846, max_tokens: 1397

prompt Log for Agent 3: YYYY-MM-DD 22:36:21,755 - log.logger_utils - INFO
You are given various scenarios involving numerical problems related to quantities, collections, and arithmetic operations. Your task is to answer questions based on the provided information for each scenario. Pay attention to details such as initial amounts, changes over time or circumstances (e.g., additions or subtractions), and specific requests regarding total counts. Use basic mathematical reasoning and calculations when determining your answers.

Answer by Agent 3: YYYY-MM-DD 22:36:24,662 - log.logger_utils - INFO
To find out how many bags of cans Tiffany had altogether, we need to focus only on the bags of cans.
On Monday, Tiffany had 3 bags of cans.
The next day, she found 7 more bags of cans.
Now, we can calculate the total number of bags of cans:
Total bags of cans = Bags on Monday +

Bags found on Tuesday
Total bags of cans = 3 + 7 = 10
Therefore, Tiffany had a total of 10 bags of cans altogether.

Agent 3 focuses on generating simple and straightforward arithmetic word problems, usually centered on addition or subtraction, with scenarios drawn from everyday life for easy understanding. The problems and solutions are very concise, typically containing only key information without extra explanations or reasoning steps. This approach is suitable for foundational arithmetic instruction or quick practice, emphasizing the calculation itself rather than the problem-solving process.

Training for Agent 4 in Interaction 1

Prameter Log for Agent 4: YYYY-MM-DD 22:36:24,768 - log.logger_utils - INFO
temperature: 0.5298405602481979,
top_p: 0.7372076916536928,
frequency_penalty: 0.02248299928407254,
presence_penalty:
0.049040988666425915,
max_tokens: 1746

prompt Log for Agent 4: YYYY-MM-DD 22:36:37,855 - log.logger_utils - INFO
Create a word problem involving everyday scenarios that require basic arithmetic operations to solve. The scenario should include quantities related to common activities such as shopping, collecting items, or completing tasks over a period of time. Ensure the question asks for a total count based on additions and potential subtractions, utilizing numbers that are simple and relatable while encouraging mathematical reasoning in interpreting the context provided.

Answer by Agent 4: YYYY-MM-DD 22:36:41,333 - log.logger_utils - INFO
To find out how many bags of cans she had altogether, we only need to focus on the bags of cans.
1. Start with the number of bags of cans Tiffany had on Monday: 3 bags.
2. Add the number of bags of cans she found on Tuesday: 7 bags.
Now, calculate the total number of bags of cans:

$$3 \text{ bags (Monday)} + 7 \text{ bags (Tuesday)} = 10 \text{ bags of cans.}$$

Therefore, Tiffany had 10 bags of cans altogether.
(Note: The bags of bottles are not included in the total for cans.)

Agent 4's prompts emphasize problem generation that incorporates real-life scenarios, encouraging mathematical reasoning and the filtering of relevant information. Its answers are usually broken down into steps, highlighting the problem-solving approach and information extraction, but do not directly provide the final answer, focusing more on guiding the Agent to think independently. The overall style leans towards instructional guidance, helping the Agent understand the problem and develop independent problem-solving skills.

Training for Agent 5 in Interation 1

Prameter Log for Agent 5: YYYY-MM-DD 22:36:41,435 - log.logger_utils - INFO
temperature: 0.3603354498081722,
top_p: 0.8427861799350946,
frequency_penalty: 0.0572741656361528,
presence_penalty: -0.124744427417488,
max_tokens: 1736

prompt Log for Agent 5: YYYY-MM-DD 22:36:53,728 - log.logger_utils - INFO
Create a problem-solving question that involves basic arithmetic operations such as addition, subtraction, multiplication, or division within various contexts. The scenarios should include everyday situations involving quantities of objects or items to be calculated. Make sure the questions require an understanding of how to combine different amounts and solve for totals or differences based on the information provided in each scenario.

Answer by Agent 5: YYYY-MM-DD 22:37:00,748 - log.logger_utils - INFO
To solve the problem, we need to focus only on the bags of cans, as the question specifically asks for the total number of

bags of cans Tiffany had.
1. On Monday, Tiffany had 3 bags of cans.
2. On Tuesday, she found 7 more bags of cans.
Now, we can add the number of bags of cans she had on Monday to the number she found on Tuesday:

$$\text{Total bags of cans} = \text{Bags on Monday} + \text{Bags found on Tuesday}.$$

Substituting the values:

$$\text{Total bags of cans} = 3 + 7$$

Now, we perform the addition:

$$\text{Total bags of cans} = 10$$

Thus, Tiffany had a total of 10 bags of cans altogether.

Agent 5's prompts cover a variety of operations, including addition, subtraction, multiplication, and division, with a strong emphasis on information integration and analytical skills. Its solutions are very detailed, systematically presenting the problem-solving process step by step, including formulas, substitution, calculation, and conclusion, with formal and well-organized language.

A.6.2 Evaluating Examples

This section systematically evaluates the performance of the agent during a specific task iteration. The evaluation process adopts a structured scoring system, conducting detailed analysis and scoring across three core dimensions: logic, ingenuity, and completeness, ultimately resulting in a comprehensive total score.

Evaluating for Agent 1 in Interation 1

Evalauation result: YYYY-MM-DD 22:37:10,792 - log.logger_utils - INFO

Logic (0 - 100):
"The solution follows a clear logical structure. It begins with understanding the question and identifying key data points, followed by performing the correct mathematical operations to arrive at the answer. The steps are clearly articulated, ensuring that the reasoning is easy to follow."
(Score: 90)

Ingenuity (0 - 100):
"While the solution is logical and methodical, it lacks any innovative or creative approaches to solving the problem. It simply adheres to standard arithmetic operations without introducing any unique methods or insights."
(Score: 70)

Completeness (0 - 100):
"The answer comprehensively addresses all parts of the question, including reading the question, identifying numerical data, performing calculations, and verifying the final result. However, it could benefit from a brief summary reiterating the significance of reaching this total within the context."
(Score: 85)

Total Score (0 - 100): (Score: 82)

Specifically, the evaluator first reviews the agent's solution process to determine whether it follows clear logical steps, including understanding the problem, extracting key data points, and performing correct mathematical calculations; based on this, a score is given for the logic dimension. Next, the evaluator examines whether the agent demonstrates unique or innovative approaches in problem-solving, or simply relies on conventional methods, and assigns a score for ingenuity accordingly. Finally, the evaluator checks whether the answer fully addresses all aspects of the problem, including data identification, calculation, and result verification, as well as whether the significance of the result is summarized, thus determining the completeness score. Each dimension is accompanied by a brief written assessment highlighting strengths and weaknesses, and a comprehensive total score is calculated based on an overall consideration. This evaluation method provides a holistic reflection of the agent's problem-solving performance and offers concrete references for subsequent optimization and improvement.

Evaluating for Agent 2 in Interaction 1

Evalauation result: YYYY-MM-DD 22:37:19,822 - log.logger_utils - INFO

Logic (0 - 100):
"The answer correctly identifies the number of bags of cans Tiffany had by logically breaking down the information provided. It accurately sums the bags from Monday and Tuesday without including irrelevant details about bottles."
(Score: 95)

Ingenuity (0 - 100):
"The approach is straightforward and effective, but it does not demonstrate any creative or innovative problem-solving techniques. The solution follows a standard arithmetic process without additional insights."
(Score: 70)

Completeness (0 - 100):
"The answer addresses the question fully by providing a clear calculation and stating that bags of bottles are not included in the total for bags of cans. However, it could have briefly summarized why only cans were considered to enhance clarity."
(Score: 85)

Total Score (0 - 100): (Score: 83)

Evaluating for Agent 3 in Interation 1

Evalauation result: YYYY-MM-DD 22:37:27,859 - log.logger_utils - INFO

Logic (0 - 100):
"The solution correctly identifies the relevant information and follows a clear logical flow to arrive at the answer. The calculations are straightforward and accurate."
(Score: 90)

Ingenuity (0 - 100):
"The solution employs basic arithmetic to solve a simple problem. While it is effective, there is little creativity or innovation in the approach."
(Score: 60)

Completeness (0 - 100):
"The answer thoroughly addresses the problem, providing all necessary calculations and arriving at a final total. However, it could be enhanced by including a brief summary of the given information."
(Score: 85)

Total Score (0 - 100): (Score: 78)

Evaluating for Agent 4 in Interation 1

Evalauation result: YYYY-MM-DD 22:37:37,127 - log.logger_utils - INFO

Logic (0 - 100):
"The reasoning presented in the answer is clear and follows a logical sequence. It identifies the relevant information (the number of bags of cans) and correctly excludes irrelevant information (the bags of bottles). The calculation is straightforward and accurate."
(Score: 90)

Ingenuity (0 - 100):
"The answer demonstrates a basic understanding of arithmetic and problem-solving. However, it lacks creative or innovative approaches to the problem, as it simply applies basic addition without any additional insights or unique methods."
(Score: 50)

Completeness (0 - 100):
"The answer provides a complete solution to the problem by addressing all necessary steps to arrive at the final count of bags of cans. It states clearly what is included in the count and what is not, making it easy to follow."
(Score: 90)

Total Score (0 - 100): (Score: 76)

Evaluating for Agent 5 in Interation 1

Evalauation result: YYYY-MM-DD 22:37:37,229 - log.logger_utils - INFO

Logic (0 - 100):
"The solution follows a clear and logical sequence of steps to arrive at the answer. It correctly identifies the relevant information, uses appropriate arithmetic, and clearly states the final result."
(Score: 95)

Ingenuity (0 - 100):
"The approach is straightforward and relies on basic arithmetic, which does not showcase high levels of creativity or innovative problem-solving. However, it effectively solves the problem without unnecessary complexity."
(Score: 70)

Completeness (0 - 100):
"The answer provides a complete solution by including all necessary calculations and explanations. It explicitly states the initial quantities, performs the addition, and presents the final answer clearly."
(Score: 90)

Total Score (0 - 100): (Score: 85)

A.6.3 Optimizing Examples

This section illustrates the process of parameter updating during a particular iteration. Specifically, the system first ranks the agents participating in the current round of tasks and selects the top three performers. Next, the system updates the parameters of a certain agent (usually one that performed well or needs improvement). The parameter update involves adjusting key hyperparameters such as temperature, top_p, frequency_penalty, and presence_penalty, all of which directly affect the diversity, creativity, and accuracy of the agent's generated responses. The system records the specific values of these parameters before and after the update, and provides a task prompt to clarify the type of problem the agent needs to solve and any important considerations.

Ranking for Iteration 1

Ranking: YYYY-MM-DD 22:37:54,082 - log.logger_utils - INFO

Top three results from message processing LLM:
"First Place": "2",
"Second Place": "5",
"Third Place": "1"

Updating for Agent in Iteration 1

Before Updating: YYYY-MM-DD 22:37:54,082 - log.logger_utils - INFO
Parameter:
temperature=0.9193923355128871,
top_p=0.9027935433165666,
frequency_penalty=1.8091973063219,
presence_penalty=-1.7443228132190376
prompt:
You are given various scenarios involving numerical problems related to quantities, collections, and arithmetic operations. Your task is to answer questions based on the provided information for each scenario. Pay attention to details such as initial amounts, changes over time or circumstances (e.g., additions or subtractions), and specific requests regarding total counts. Use basic mathematical reasoning and calculations when determining your answers.

After Updating: YYYY-MM-DD 22:38:02,220 - log.logger_utils - INFO
Parameter:
temperature=0.41004817918398145,
top_p=0.9247874518938444,
frequency_penalty=-0.0035787094690344773,
presence_penalty=0.025675460113628396
prompt:
You are presented with various scenarios that involve numerical problems related to quantities and arithmetic operations. Your task is to answer questions based on the provided information for each scenario. Start by carefully reading each question to understand what is being asked. Identify all relevant numerical data and quantities mentioned. Determine which mathematical operations (addition, subtraction, multiplication, division) are needed based on keywords or phrases that indicate relationships between numbers. Perform calculations step-by-step, keeping track of intermediate results if necessary, and double-check your final answer against the context of the problem to ensure it makes sense. Pay attention to details such as initial amounts, changes over time, and specific requests regarding total counts, using basic mathematical rea-

soning and calculations to determine your answers.

This entire process embodies an automated "evaluation–selection–fine-tuning–re-evaluation" optimization loop, aiming to continuously improve the agent's performance on specific types of problems through iterative trial and adjustment.

B Responsible Use Statement

Reproducibility and transparency. We are committed to the full reproducibility of this work. The proposed Grey Wolf Optimizer (GWO)-based multi-agent collaboration framework, along with its core algorithms and iterative optimization procedure, is fully described with pseudocode in the Appendix to ensure that researchers can directly reproduce and extend our study. Our experimental setup is detailed in Section 4. We evaluate the framework on both mathematical reasoning and hybrid reasoning benchmarks. These publicly available datasets cover arithmetic, algebra, geometry, logic, and interdisciplinary reasoning, thereby ensuring diverse experimental validation. For model selection, we employ several mainstream large language models to verify the robustness of our method across different scales and architectures. To guarantee transparency, the algorithm pseudocode (including the standard GWO and the agent-based iterative optimization process) is provided in Appendix A, while experimental results, ablation studies, and additional performance comparisons are presented in Appendix A.4 and A.5. All experiments are implemented in a Python environment and executed on a multi-GPU system to ensure efficiency in both inference and training. We will release the complete source code and configuration files upon publication, enabling other researchers to directly verify and extend our results.

THE USE OF LARGE LANGUAGE MODELS (LLMS) The authors utilized OpenAI's GPT-5 to improve the grammar, clarity, and conciseness of the text. All scientific contributions, methodology design, experiments, and analyses are the original work of the authors, who take full responsibility for the paper's content.