TOWARDS EFFICIENT AND SCALABLE MULTI-AGENT REASONING VIA BAYESIAN NASH EQUILIBRIUM

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) exhibit strong reasoning capabilities, which can be further enhanced through multi-agent frameworks. However, existing multi-agent methods often suffer from high computational costs and lack theoretical convergence guarantees. To address these limitations, we introduce an incomplete information perspective to enhance the scalability of multiple LLMs by modeling them with Bayesian Nash Equilibrium (BNE) and propose **E**fficient **Co**ordination via **Nash** Equilibrium (EcoNash), a hierarchical reinforcement learning framework. EcoNash guides multiple LLMs to achieve BNE by integrating distributed reasoning and centralized commitment, ensuring that each LLM independently generates optimal answers based on its own beliefs without the need for extensive inter-agent communication. Theoretically, we prove that our framework achieves a regret bound of $O\left(N\sqrt{T}/1-\gamma\right)$, which grows sublinearly with T, while multi-agent frameworks that do not attain BNE can at best achieve $O\left(\delta_{\max}T/1-\gamma\right)$. Empirically, our method outperforms single-LLM approaches by 10.9% and surpasses existing multi-LLM methods by 11.2% over six benchmark tests covering complex reasoning and planning tasks on average. Additionally, scalability experiments show that our approach efficiently integrates more models, confirming its flexibility and scalability, potentially leading to larger multi-LLM ensembles.

1 INTRODUCTION

Large Language Models (LLMs) (Brown et al., 2020) have demonstrated exceptional reasoning capabilities across various tasks, including natural language understanding, generation, and complex problem-solving. Recent research enhances their reasoning abilities by exploring multi-agent frameworks (Du et al., 2024; Chan et al., 2024; Liang et al., 2023; Chen et al., 2023; Hong et al., 2023) where multiple LLMs collaborate. These frameworks simulate human-like discussions, boosting diversity and creativity and potentially yielding more robust solutions in real-world applications.

However, existing multi-agent frameworks are computationally expensive, as they require multiple model instances and repeated rounds of interaction (Wu et al., 2023). Agents must read and process one another's outputs, increasing communication overhead and latency. Adding components such as judges or verifiers further compounds the problem by introducing more computational layers (Zheng et al., 2023). What's more, the current multi-agent debate (MAD) systems lack theoretical guarantees for convergence(Du et al., 2024), while MAD between LLMs can be viewed as games that need to converge to a single, stable solution. While empirical results may demonstrate convergence in certain cases, the introduction of a judge can further guide the debate direction(Lu et al., 2024), the lack of solid theoretical foundations leaves the reliability and stability of such systems uncertain.

To address the above challenges, we propose a novel framework called *EcoNash* (*E*fficient *Co*ordination via *Nash* Equilibrium), which introduces a *Bayesian Nash Equilibrium* (*BNE*) perspective to multi-LLM systems. Inspired by reinforcement learning, our framework constructs a hierarchical coordination mechanism. Each Execution LLM operates independently with its own belief network, receiving only the question and strategy from the Coordinator LLM. This enables multiple Execution LLMs to engage in distributed reasoning, guided by the Coordinator LLM, to achieve BNE by optimizing the belief network and belief encoder. Optimization employs adaptive rewards and an early stopping criterion. When the outputs of the Execution LLMs consistently meet convergence metrics, the system is considered to have reached an approximate BNE, and further iterations are halted. This approach not only reduces unnecessary computations but also minimizes the input tokens required by the Coordinator LLM, enhancing overall efficiency. Unlike existing methods, Execution LLMs can generate outputs in parallel without the need for extensive inter-agent communication in EcoNash, reducing both communication costs and computational overhead.

Theoretically, we demonstrate that EcoNash achieves a regret bound of $O\left(N\sqrt{T}/1-\gamma\right)$, which grows sublinearly with T. In contrast, multi-agent frameworks that do not attain BNE can at best achieve a regret bound of $O\left(\delta_{\max}T/1-\gamma\right)$. Our framework's convergence toward BNE provides strong theoretical guarantees for efficiency, while inference incurs lower consumption costs than existing multi-LLM systems, providing significant insights for scaling up multi-LLM systems. Based on it we verify whether EcoNash can address scalability, a challenge often overlooked in prior works (Wu et al., 2024; Yin et al., 2023; Lan et al., 2024; Yuan et al., 2024a). By constructing a Coordinator-Execution subsystem based on local Nash equilibria, we scale EcoNash to a larger LLMs ensemble framework (Central-Coordinator-Execution) in global Nash, resulted in enhanced performance.

Through extensive experiments on six benchmarks, including complex reasoning and planning tasks, our method outperforms single-agent approaches by 10.9% and surpasses the performance of existing multi-agent methods by 11.2% in average, confirming the robustness and efficiency of our framework. Scalability experiments further demonstrate that EcoNash effectively integrates numerous models, showcasing its applicability in large-scale settings. When the number of Execution LLMs is increased to nine, performance improves by 18.1% compared to three Execution LLMs.

We summarize our major contributions as follows:

- We conceptually formalize BNE in multi-LLM systems and technically instantiate it through a hierarchical optimization framework *EcoNash* to improve reasoning over collaboration of LLMs.
- We address the non-trivial challenge of scaling up multi-LLM systems with local-global Nash, facilitated by EcoNash's low reliance on inter-agent communication and convergence guarantee.
- Extensive experiments on six benchmarks demonstrate that EcoNash outperforms existing single- and multi-agent methods, while scalability experiments confirm its ability to efficiently integrate numerous models for large-scale settings, potentially leading to larger multi-LLM ensembles.

2 Related Work

Prompting Large Language Models to Reason. Large language models are significantly more capable of complex reasoning with the advancement of prompt techniques (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023; Yao et al., 2023; Chia et al., 2023; Fu et al., 2022; Wan et al., 2023; Zhang et al., 2023b; Zhou et al., 2022). Wei et al. (2022) introduced Chain-of-Thought (CoT) prompting, which presents step-by-step reasoning examples within the prompt. This enables the model to engage in explicit reasoning, enhancing its ability to follow the logical progression that leads to the correct answer. Various extensions of CoT have been developed to improve reasoning performance further. Zero-shot CoT (Kojima et al., 2022) eliminates the need for manually constructing exemplars, prompting models with phrases like "Let's think step by step" to encourage reasoning. Wang et al. (2023) proposed self-consistency (SC) sampling, where multiple reasoning paths are sampled, and the final answer is determined by majority voting. To enable LLMs to engage in deliberate decision-making, Tree of Thoughts (ToT) Yao et al. (2023) generates multiple potential answers at each reasoning step, building a tree of possible solutions. It then applies breadth-first or depth-first search to navigate the tree, ultimately determining the rationale and final answer.

Multi-agent Debate for Large Language Models Reasoning. Various multi-agent debate strategies(Du et al., 2024; Chan et al., 2024; Liang et al., 2023; Chen et al., 2023; Smit et al., 2024; Zhang et al., 2023a; Pham et al., 2023) have been developed to strengthen the reasoning ability of LLMs further. Du et al. (2024) introduced an approach where multiple instances of LLMs propose their individual reasoning processes, engaging in multiple rounds of debate to reach a consensus on the final answer. This method not only significantly enhances reasoning performance across a variety of tasks but also reduces the occurrence of hallucinations. Some studies(Chan et al., 2024; Liang et al., 2023) incorporate role-playing into multi-agent debate strategies using role-specific prompts, which foster divergent thinking and enhance the reasoning capabilities of LLMs. However, current multi-agent debate strategies face high computational costs and lack theoretical guarantees for convergence. In this work, we introduce an incomplete information perspective to enhance the scalability of multiple LLMs to ensure independent reasoning by each Execution LLM while addressing communication cost. Our framework ensures convergence through rigorous theoretical analysis.

3 METHOD

In this section, we develop a theoretical framework for multi-LLM systems to achieve BNE. We begin by defining and establishing the implementation of BNE within a multi-LLM system (Section 3.1). We conduct a convergence analysis and evaluate regret bounds to demonstrate the efficiency of our method (Section 3.2). Then, we outline our optimization approach with prompt embeddings (Section 3.3), integrating both inference and optimization processes in Section 3.3 , followed by our scaling-up method to enhance the framework's scalability in Appendix A.4).

3.1 BAYESIAN NASH EQUILIBRIUM IN THE MULTI-LLM FRAMEWORK

3.1.1 DEFINITION AND IMPLEMENTATION OF BNE

A **Bayesian Nash Equilibrium** (BNE) is a strategy profile where each agent maximizes its expected utility based on its beliefs about other agents' strategies. In the context of incomplete information games, where each LLM does not have direct access to the outputs of other LLMs, we construct a hierarchical framework consisting of execution LLMs and a coordinator LLM to establish the game. The coordinator LLM takes a question as input and outputs corresponding strategy and format specifications to guide execution LLMs. After receiving answers from execution LLMs, it generates a final commitment to address the question. Each execution LLM maintains its belief state $\mathbf{b}_i \in \mathbb{R}^d$ and receives observations $O_i = [e_t, e_s, \mathbf{b}_i]^\top$, where e_t encodes the task and e_s represents the coordinator's strategy. To enable coordination without direct information sharing, we implement a belief network $B_i(\tau_i, O_i; \theta_i^B)$ that updates each agent's state based on its history τ_i and current observation, generating prompt embeddings $\mathbf{e}_i$. A belief encoder $f_e(\{\mathbf{b}_i\}_{i=1}^N; \theta_e)$ then aggregates these beliefs into group information $\mathbf{E}$, and then the centralized mixing network of coordinator LLM processes this group information to guide coordination through a commitment C.

To quantify the effectiveness of different belief states, we employ Q-functions $Q_i(O_i, \mathbf{e}_i; \theta_i^B)$ that evaluate prompt embeddings generated by the belief network. These value estimates guide the optimization of belief network parameters θ_i^B. A BNE is achieved when each agent's belief network parameters generate prompt embeddings that maximize its expected utility:

$$\mathbf{e}_i^* = \arg\max_{\mathbf{e}_i} \mathbb{E}_{\mathbf{E}\sim f_e(\{\mathbf{b}_j\}_{j=1}^N;\theta_e)} \left[U_i(O_i, \mathbf{E}, \mathbf{e}_i)\right].$$

To guarantee the existence of BNE, the following conditions need to be established:

- **Compactness and Convexity**: For each agent i, the mixed strategy space Π_i is non-empty, compact, and convex, consisting of all mappings from types Θ_i to probability distributions.
- **Continuity**: The payoff function $U_i(\theta, a)$ is continuous in the type profile and the action profile.
- **Quasi-Concavity**: For each agent i, the expected payoff is quasi-concave in a_i for fixed θ_i.

Under these conditions, we can apply Glicksberg's Fixed Point Theorem (Ahmad et al., 2023) to guarantee the existence of BNE. Specifically, the best response correspondences $BR_i(\pi_{-i})$ for each agent i are non-empty, convex-valued, and upper hemicontinuous.

Theorem 1 (Existence of Bayesian Nash Equilibrium). *In the multi-agent LLM framework with the specified conditions, there exists a Bayesian Nash Equilibrium strategy profile $\overline{\pi}^* = (\pi_1^*, \pi_2^*, \ldots, \pi_N^*)$ such that no agent can unilaterally deviate to improve its expected payoff, given its beliefs about other agents' types and strategies. For the proof, please refer to Appendix A.1.*

Proposition 1 (Convergence to Bayesian Nash Equilibrium). *Under appropriate assumptions about the learning rate, exploration strategy, and Q-network properties, the prompt embedding adjustment via TD loss converges to a Bayesian Nash Equilibrium The proof is provided in Appendix A.2.*

3.2 CONVERGENCE ANALYSIS AND BAYESIAN REGRET BOUND

In this section, we analyze the convergence properties of our EcoNash framework through Bayesian regret. Our analysis demonstrates that the framework's belief network structure and coordinated learning process lead to efficient convergence toward BNE, achieving sublinear regret bound $O\left(N\sqrt{T}/1-\gamma\right)$ in contrast to the linear regret of existing multi-agent debate methods.

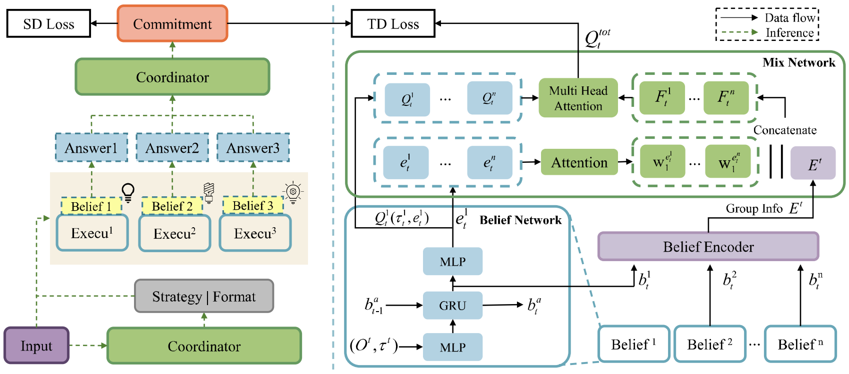

Figure 1: The EcoNash framework. The inference procedure is shown by green arrows: the coordinator receives the question, provides a strategy to the Execution LLM, which outputs an answer. Afterwards, the coordinator forms the final commitment. Simultaneously, the Execution LLM passes its belief to the belief encoder, embedding agent information. TD Loss updates the belief network, and SD Loss updates the belief encoder, optimized to achieve BNE, as the red gradient flow.

For each agent i, we measure the learning efficiency using Bayesian regret over T steps:$R_i(T) = \mathbb{E}_{s_t,\pi_t}\left[\sum_{t=1}^{T}\left(V_i^*(s_t) - V_i^{\pi_t}(s_t)\right)\right]$, where $V_i^*(s)$ represents the optimal value under BNE policies and $V_i^{\pi_t}(s)$ is the value under current policies at time t. The expectation accounts for randomness in both state transitions and policy choices. To analyze the total Bayesian regret $R(T) = \sum_{i=1}^{N} R_i(T)$, we make standard assumptions (see Appendix A.3) to propose Lemma 1, and we prove Lemma 1 in B.1. Using Lemma 1 we bound the Bayesian regret and provide a proof sketch here, with detailed proofs and comparision with multi-agent debate in Appendix B.2 and B.3.

Lemma 1 (Performance Difference). *For joint policies $\pi = (\pi_i, \pi_{-i})$ and $\pi' = (\pi_i', \pi_{-i}')$, the value difference for agent i is:*

$$V_i^{\pi'}(s) - V_i^{\pi}(s) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi'}}\left[\mathbb{E}_{a\sim\pi'}Q_i^{\pi}(s,a) - \mathbb{E}_{a\sim\pi}Q_i^{\pi}(s,a)\right],$$

where $d_{\pi'}$ is the state distribution under π', and $a = (a_i, a_{-i})$ denotes joint actions.

Applying this lemma to our regret analysis yields (Jin et al., 2020; Fujimoto et al., 2018):

$$R(T) = \sum_{i=1}^{N}\frac{1}{1-\gamma}\mathbb{E}_{s_t,\pi_t}\left[\sum_{t=1}^{T}\left(\mathbb{E}_{a_t^*\sim\pi^*}Q_i^{\pi_t}(s_t,a_t^*) - \mathbb{E}_{a_t\sim\pi_t}Q_i^{\pi_t}(s_t,a_t)\right)\right]$$

where π^* represents the BNE policies. Through analysis of estimation error ϵ_t and policy suboptimality δ_t , we establish: $\mathbb{E}_{a_t^*\sim\pi^*}Q_i^{\pi_t}(s_t,a_t^*) - \mathbb{E}_{a_t\sim\pi_t}Q_i^{\pi_t}(s_t,a_t) \leq 2\epsilon_t + \delta_t$. This leads to:

$$R(T) \leq \sum_{i=1}^{N}\frac{1}{1-\gamma}(2C_\epsilon + C_\delta)\sum_{t=1}^{T}\frac{1}{\sqrt{t}} = O\left(\frac{N\sqrt{T}}{1-\gamma}\right).$$

3.3 Framework of EcoNash

In this section, we present a framework designed to achieve BNE within a multi-LLMs system, satisfying the assumptions in Appendix A.3 to enable Lemma 1 can be applied to analyze its Bayesian regret. The framework has two primary phases: **Inference** and **Optimization**. The inference phase involves generating and propagating strategies and responses, while optimization phase focuses on updating strategies to align with global objectives and optimizes their beliefs to achieve BNE.

3.3.1 Inference Phase

During the inference phase, a Coordinator LLM generates an informative strategy and a format based on the input question q. These are then disseminated to the Execution LLMs, which independently produce their respective answers. Finally, the Coordinator LLM aggregates these answers to form a final commitment, detailed inference flow as illustrated clearly in Figure 1: the green inference flow.

3.3.2 Optimization Phase

The optimization phase of EcoNash implements a hierarchical learning framework under the centralized training with decentralized execution (CTDE) paradigm(Foerster et al., 2018b; Kraemer & Banerjee, 2016), satisfying our theoretical assumptions while optimizing towards the Bayesian Nash Equilibrium (BNE). Under Assumption 2, execution LLMs aim to align with posterior distributions determined by the coordinator LLM, achieved through our belief network architecture. The game regularity (Assumption 3) ensures stable information gain across timesteps, guiding our design of the belief encoder. The concentrability condition (Assumption 4) bounds the error in value estimation, informing our mixing network structure. The optimization procedure is summarized in Algorithm 1.

Reward Setting The reward function R is central to the optimization stage, providing feedback on each agent's performance. Multiple types of rewards are designed to capture different aspects of performance. The Action Likelihood Reward evaluates the consistency of an agent's actions with the commitment C, inspired by maximum entropy inverse reinforcement learning (Zhu et al., 2023). Task-specific rewards address correctness in tasks like math problem solving or relevance in planning (Hao et al., 2023). The Self-Evaluation Reward enables the coordinator to assess the quality of generated answers, promoting coherence, consistency, and creativity across agents, driving optimization toward BNE (Xie et al., 2024b). More details are provided in Appendix B.4.

Individual Belief Network Execution i employs a belief network $B_i(\tau_i, O_i; \theta_i^B)$ to update its belief state $\mathbf{b}_i$ based on its history trajectory τ_i and current observation O_i. The belief state $\mathbf{b}_i$ is used to adjust the prompt embedding $\mathbf{e}_i = [T_i, p_i]$, which defined as:

$$T_i = T_{\min} + (T_{\max} - T_{\min}) \cdot \sigma(W_T \mathbf{b}_i + b_T), \quad p_i = p_{\min} + (p_{\max} - p_{\min}) \cdot \sigma(W_p \mathbf{b}_i + b_p),$$

with $\sigma(\cdot)$ as the sigmoid activation function. Here, T_i adjusts the softmax distribution, and p_i sets the sampling threshold. The belief network outputs the prompt embedding $\mathbf{e}_i$ and Q-value Q_i^t for the mixing network, while passing $\mathbf{b}_i$ to the belief encoder for group-level dynamics. It is optimized using the TD loss, where r_i^t is the local reward and ϕ denotes the parameters of the Q-value function:

$$\mathcal{L}_{\text{TD}}^i(\theta_i^B) = \mathbb{E}_{\mathcal{D}} \left[\left(r_i^t + \gamma \max_{\mathbf{e}_i^{t+1}} Q_i^{t+1}(\tau_i^{t+1}, \mathbf{e}_i^{t+1}; \phi') - Q_i^t(\tau_i^t, \mathbf{e}_i^t; \phi) \right)^2 \right],$$

Belief Encoder The belief encoder $f_e(\cdot; \theta_e)$ aggregates the belief states from all agents to generate a group-level representation $\mathbf{E} = f_e(\{\mathbf{b}_i\}_{i=1}^N; \theta_e)$. using multi head attention with H attention heads to capture inter-agent relationships. Each head is computed as $\text{head}_h = \text{Attention}(W_h^Q \mathbf{b}, W_h^K \mathbf{b}, W_h^V \mathbf{b})$, and the final output is obtained by $\mathbf{E} = \text{Concat}(\text{head}_1, ..., \text{head}_H) W^O$, with W_h^Q, W_h^K, W_h^V being learnable parameters, and W^O is the output projection matrix. The belief encoder is optimized as: $\mathcal{L}_e(\theta_e) = \mathcal{L}_{\text{TD}}^{\text{tot}}(\phi) + \lambda_e \sum_i L_{\text{TD}}^i(\theta_i^B)$.

Centralized Mixing Network The Centralized Mixing Network is designed to coordinate belief information from execution LLMs, facilitating optimization towards BNE. Prompt embeddings $\{\mathbf{e}_i^t\}_{i=1}^N$ are processed via self-attention to capture intra-agent dependencies, producing transformed embeddings $\{\mathbf{w}_i^t\}_{i=1}^N$. These embeddings are concatenated with the group-level representation $\mathbf{E}^t$ to generate feature transformations $\{F_i^t\}_{i=1}^N$, encoding both local agent-specific and global group-level information. The feature transformations $\{F_i^t\}_{i=1}^N$ and individual Q-values $\{Q_i^t\}_{i=1}^N$ are then combined via multi-head attention to compute the global value function Q_{tot}^t, capturing complex local-global interactions. The network is optimized by minimizing the composite loss: $\mathcal{L}_{\text{mix}}(\phi) = \mathcal{L}_{\text{TD}}^{\text{tot}}(\phi) + \mathcal{L}_{\text{SD}} + \lambda_m \sum_i \|Q_i^t - Q_{\text{tot}}^t\|^2$, where the TD loss aligns Q_{tot}^t with r_{tot}:

$$\mathcal{L}_{\text{TD}}^{\text{tot}}(\phi) = \mathbb{E}_{\mathcal{D}} \left[\left(r_{\text{tot}} + \gamma \max_{\{\mathbf{e}_i^{t+1}\}} Q_{\text{tot}}^{t+1}(\tau_{t+1}, \{\mathbf{e}_i^{t+1}\}; \phi') - Q_{\text{tot}}^t(\tau_t, \{\mathbf{e}_i^t\}; \phi) \right)^2 \right],$$

with $\tau_t = \{O_i^t\}_{i=1}^N$ representing the joint observations, and $\{\mathbf{e}_i^t\}_{i=1}^N$ as the agents' belief embeddings. The similarity difference (SD) loss aligns the feature transformations $\{F_i^t\}_{i=1}^N$ with the coordinator LLM's commitment C: $\mathcal{L}_{\text{SD}} = \lambda_b \sum_i (1 - \text{sim}(F_i^t, C))^2$. A consistency term further ensures Q_i^t aligns with Q_{tot}^t. The target parameters ϕ' are updated via a soft update rule: $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$, where τ is the update rate. By synthesizing belief information and aligning with C, the mixing network ensures monotonicity, guaranteeing that improvements in individual agent performance positively impact global coordination, enabling stable convergence to the equilibrium. The detailed proof of monotonicity can be found in Appendix A.5.

EARLY STOPPING To ensure efficient optimization and convergence to stable solutions, early stopping is implemented based on three key criteria. First, Commitment Stability is achieved when the change in the coordinator's commitment satisfies $\|\Delta C\| = \|C_{t+1} - C_t\| \leq \epsilon_C$. Second, Reward Convergence is monitored such that the average reward across agents reaches a predefined threshold, $\frac{1}{N}\sum_{i=1}^{N} r_i \geq R_{\text{threshold}}$. Lastly, Loss Convergence is ensured when the total loss stabilizes, satisfying $|L_{\text{tot}}^{t+1} - L_{\text{tot}}^{t}| \leq \epsilon_L$, where L_{tot} is the sum of individual agent losses $\sum_i L_i$, execution loss L_e, and the mixing loss L_{mix}. These criteria comprehensively monitor the optimization process, ensuring both strategic alignment and task performance while preventing premature termination.

Algorithm 1 Optimization Phase of EcoNash

```
Require: Execution LLMs {ExecLLM_i}, Coordinator LLM, Networks {f_e, f_mix}
Require: Thresholds {ε_C, R_threshold, ε_L}, Maximum episodes T_max
Ensure: Optimized network parameters
 1: while not converged and t < T_max do
 2:     // Parallel execution and local optimization for each agent
 3:     for each Execution LLM i do
 4:         Update belief state b_i and generate output u_i          ▷ Run execution LLM
 5:         Compute rewards: r_i ← α_1 r_i^AL + α_2 r_i^TS + α_3 r_i^SE   ▷ Action likelihood + Task +
        Self-evaluation
 6:         Store transition (O_i, u_i, r_i, O'_i) in replay buffer D
 7:         Update individual belief network parameters             ▷ Using TD loss
 8:     end for
 9:     // Global coordination and optimization
10:     Update belief encoder f_e                     ▷ Using global TD loss + local TD losses
11:     Update mixing network f_mix                   ▷ Using TD + similarity + consistency losses
12:     Get new commitment C_{t+1} from Coordinator
13:     // Check convergence conditions
14:     if ||C_{t+1} − C_t|| ≤ ε_C and R_avg ≥ R_threshold and |L_tot^{t+1} − L_tot^t| ≤ ε_L then
15:         break                                     ▷ Early stopping when all criteria are met
16:     end if
17: end while
```

4 EXPERIMENT

In this section, we present the experiment setup in Section 4.1, demonstrate the performance against baseline methods in Section 4.2, validate the heterogeneous results of different models in Section 4.3, test scale-up capability in Section 4.4, and conduct ablation studies in Section 4.5.

4.1 SETUPS

Models and Datasets. We evaluate 4 newly released opensourced LLMs: `LLaMA3.1 8B` (Dubey et al., 2024), `LLaMA3.1 70B`, `Mistral-7B` (Jiang et al., 2023), `LLaMA3.1 405B` across 5 reasoning tasks, including 4 mathematical tasks (GSM8K (Cobbe et al., 2021), GSM-Hard (Gao et al., 2023), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021)) and one commonsense reasoning task (StrategyQA (Geva et al., 2021)). Then, we evaluate the most powerful LLM (GPT4 turbo) in a very challenging planning task (Travelplanner (Xie et al., 2024a)) to further validate the performance. The details of evaluation tasks can be found in Appendix B.5.

Compared Methods and Evaluation Metrics We compare EcoNash against several strong baseline types widely adopted: (i) single-round CoT prompting, including zero-shot and few-shot CoT (Kojima et al., 2022; Wei et al., 2022); (ii) multi-round CoT prompting, Self Consistency SC (Wang et al., 2023) method, where we sample answers 64 times and employ majority voting for answer selection; (iii) value-guided search approaches with learned action-value functions, including TS-LLM (Feng et al., 2023) which leverages AlphaZero-style value networks for MCTS, and PPO-MCTS (Liu et al., 2024) which learns value models to evaluate generation quality in tree search; (iv) multi-round self-improving approaches, using ToT (Yao et al., 2023), RAP (Hao et al., 2023) and React(Yao et al., 2022) as baselines, with BFS and MCTS for tree search, respectively, following their original implementations for answer selection; and (v) multi-LLM reasoning frameworks, including rStar (Qi et al., 2024) and multi-agent debate (Du et al., 2024).

EcoNash Setups In this section, the EcoNash framework includes one coordinator and three Execution LLMs. The hyperparameters for training can be found in Appendix B.6. To ensure a fair comparison with the baseline, we use four identical models for these LLMs. For heterogeneous results, we also evaluate EcoNash with different models in Table 3. All evaluations are conducted in a zero-shot setting, with a general prompt provided in Appendix C. Notably, while we set a 50-token constraint for the coordinator's strategy generation, considering that LLMs may not strictly follow length instructions (Yuan et al., 2024b), who showed that 95% of responses stay within 1.4× and 50% within 1.0× of the specified length, we implement a 70-token hard cutoff with regeneration mechanism, which effectively controls the token usage as verified in Table 4.

4.2 Main Result

Table 1 shows a detailed comparison of each method on four mathematical and one commonsense reasoning dataset. Empirical results demonstrate that EcoNash outperforms most baselines across all complex reasoning benchmarks. On average, EcoNash outperforms the single-round method Zero-shot CoT by 25.6%, Few-shot CoT by 6.3%, multi-round CoT prompting SC by 10.9%, multi-round self-improving approaches ToT by 11.2%, multi-LLM reasoning frameworks rStar by 6.4%.

Furthermore, when evaluated on the very challenging Travelplanner benchmark using GPT-4-Turbo in Table 2, EcoNash enhanced the final pass rates to 7.2% on the validation set and 9.3% on the test set, while compared to 2.3% and 3.7% achieved by a three-round multi-agent debate approach.

These results demonstrate that EcoNash effectively leverages the capabilities of more powerful models and outperforms alternative reasoning optimization methods in complex tasks. Additionally, we provide a corresponding example for MATH which are available in Appendix D. Note that EcoNash uses fewer tokens compared to multi-round CoT prompting SC, multi-round self-improving approaches ToT, and Multi-Agent Debate, meanwhile achieved performance improvements.

4.3 Additional Result

To evaluate the impact of both the Coordinator LLM and Execution LLM performance on the EcoNash framework and find whether heterogeneous Execution LLMs can also achieve a BNE, we conducted two types of experiments: one pairing a strong Coordinator LLM with weaker Execution LLMs, and another pairing a weak Coordinator LLM with stronger Execution LLMs. These experiments were further divided into homogeneous and heterogeneous execution groups for detailed analysis. To ensure a fair comparison, the Coordinator LLM was consistently set to `Llama3.1 70b` across all experiments. For the heterogeneous execution group, we used the following configurations: `Llama 3.1 8b, Llama 3 8b, and Mixtral 7b`, as well as another configuration consisting of `Mixtral 8×22b, Qwen1.5 110b, and Llama3.1 405b`. For the homogeneous execution group, two configurations were tested: one with three weak models`Llama 3.1 8b`), and another with three strong models `Llama 3.1 405b`. Experimental results indicate that stronger Execution LLMs improve performance by providing higher-quality answers and achieving BNE more efficiently. Additionally, heterogeneous model perform worse than homogeneous models due to increased challenges in reaching BNE, but still outperform baseline method Few-shot CoT .

To assess the cost efficiency of the EcoNash framework, Table 4 presents the average token usage of EcoNash, Multi-Agent Debate, RAP, and Self Consistency (SC) across the Math, GSM8K, and GSM-Hard datasets for three models: `Llama 3.1 70B`, `Mixtral 8x7B`, and `Mixtral 8x22B`. The results demonstrate that EcoNash reduces token consumption by an average of 21.4% compared to Multi-Agent Debate (3 rounds). Notably, when the Coordinator LLM provides detailed strategies with answer(as shown in the token consumption data in Table 4), token usage increases an average of 112% higher token consumption as each Execution LLMs must process the full strategy.

4.4 Scale Up Result

We analyzed the impact of varying the number of agents further to validate EcoNash across a broader range of LLMs. We conducted three sets of experiments on the MATH, GSM-Hard, SVAMP, and

Table 1: Empirical results of five reasoning datasets: GSM8K,GSM-Hard, SVAMP, Strategy QA, MATH. **Bold face** numbers indicate the best performance, while <u>underline</u> means the second best.

Dataset	Method	Mistral-8×7B	Mistral-8×22B	LLaMA3.1-70B	LLaMA3.1-405B	Average
GSM8K	Zero-shot CoT	62.06	72.14	78.38	86.40	74.74
	Few-shot CoT	74.92	84.05	95.10	<u>96.80</u>	<u>87.71</u>
	SC@maj64	71.08	<u>86.24</u>	89.56	92.40	84.82
	rStar	<u>75.82</u>	81.92	91.13	94.16	85.76
	ToT	71.46	82.60	84.52	92.73	82.83
	RAP	72.03	76.97	81.33	92.14	80.62
	TS-LLM	74.21	84.68	<u>94.82</u>	96.42	87.53
	PPO-MCTS	73.45	82.76	92.24	94.85	85.83
	EcoNash	**76.97**	**88.20**	**96.70**	**98.80**	**90.17**
GSM-Hard	Zero-shot CoT	21.47	32.24	36.78	42.17	33.17
	Few-shot CoT	26.71	41.35	45.21	52.88	41.54
	SC@maj64	22.47	<u>44.19</u>	39.76	47.39	38.45
	rStar	20.21	37.91	<u>49.82</u>	52.75	40.17
	ToT	24.39	41.71	37.25	46.84	37.58
	RAP	22.47	42.79	38.97	46.44	37.67
	TS-LLM	**26.85**	42.92	47.76	<u>55.24</u>	<u>41.69</u>
	PPO-MCTS	24.86	40.12	44.53	53.42	40.73
	EcoNash	<u>25.76</u>	**47.58**	**51.43**	**60.10**	**46.22**
SVAMP	Zero-shot CoT	81.57	86.27	85.70	91.40	86.24
	Few-shot CoT	<u>86.42</u>	91.73	94.50	<u>96.30</u>	<u>92.24</u>
	SC@maj64	83.57	88.37	93.80	95.60	90.34
	rStar	84.69	86.40	92.15	<u>95.90</u>	89.79
	ToT	83.31	89.87	88.60	93.50	88.82
	RAP	85.64	<u>91.90</u>	84.50	90.70	88.19
	TS-LLM	83.25	89.82	93.92	94.24	90.81
	PPO-MCTS	85.24	89.76	93.15	94.82	90.74
	EcoNash	**87.79**	**92.27**	**96.80**	**97.20**	**93.52**
StrategyQA	Zero-shot CoT	55.13	67.91	75.21	78.56	69.20
	Few-shot CoT	62.79	82.38	82.57	85.30	78.26
	SC@maj64	65.45	81.27	79.33	82.07	77.03
	rStar	68.64	<u>86.70</u>	83.45	87.86	<u>81.66</u>
	ToT	**71.29**	84.49	80.15	84.17	80.03
	RAP	69.38	82.27	83.29	87.92	80.72
	TS-LLM	68.12	83.82	<u>84.24</u>	<u>90.46</u>	81.65
	PPO-MCTS	67.85	82.94	83.76	89.24	80.95
	EcoNash	<u>70.21</u>	**88.27**	**87.39**	**94.30**	**85.04**
MATH	Zero-shot CoT	25.17	54.17	68.24	73.82	55.35
	Few-shot CoT	33.38	66.45	74.41	80.30	63.64
	SC@maj64	31.58	62.21	67.39	78.25	59.86
	rStar	**37.89**	<u>70.28</u>	71.57	83.49	65.81
	ToT	34.35	65.22	60.41	82.88	60.72
	RAP	33.99	62.53	68.71	80.23	61.37
	TS-LLM	34.82	67.85	<u>76.92</u>	<u>83.76</u>	<u>65.84</u>
	PPO-MCTS	34.76	65.82	73.45	81.24	63.82
	EcoNash	<u>37.02</u>	**72.29**	**81.47**	**87.50**	**69.07**

StrategyQA datasets, aiming to address three key questions: (1) To what extent can weaker LLMs be enhanced? (examined on `LLaMA 3.1 8B`), (2) Can stronger LLMs be further improved? (using `LLaMA 3.1 70B`), and (3) Should the number of Coordinator LLMs be increased along with the number of Execution LLMs? Starting from three Execution LLMs (as in the main results), we gradually increased the number of agents to nine. We used the few-shot CoT as the baseline (in grey line) as Figure2. The results suggest that beyond four Execution LLMs, performance improvements were minimal, and in some cases, performance even declined. We attribute this to the challenge faced by the Coordinator LLM in managing an excessive number of Execution LLMs, making it difficult to achieve optimal coordination by redundant information from the additional agents.

Instead of simply increasing the number of Execution LLMs, we enhance scalability by forming a global Nash equilibrium through local Nash equilibria by introducing additional coordinators. This setup ensures that each Coordinator handles a reasonable amount of data. Specifically, each Coordinator manages up to 4 Execution LLMs, forming commitments and guiding them toward local Nash equilibria. Furthermore, a central LLM was introduced to coordinate the multiple coordinators, facilitating the transition from local Nash equilibria to a global Nash equilibrium (details in Appendix 2). We observed significant improvements across all benchmarks, both for weaker models(`Llama 3.1 8B`) and stronger models (`Llama 3.1 70B`). Compared to a system with 3

Table 2: Empirical results on the TravelPlanner dataset, along with some leaderboard rankings, are presented. The best performance is highlighted in bold.

	Validation (#180)						Test (#1,000)					
	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro			Micro	Macro	Micro	Macro	
Greedy Search	100	74.4	0	60.8	37.8	0	100	72.0	0	52.4	31.8	0
Two-stage												
Mixtral-8x7B-MoE	49.4	30.0	0	1.2	0.6	0	51.2	32.2	0.2	0.7	0.4	0
Gemini Pro	28.9	18.9	0	0.5	0.6	0	39.1	24.9	0	0.6	0.1	0
GPT-3.5-Turbo	86.7	54.0	0	0	0	0	91.8	57.9	0	0.5	0.6	0
GPT-4-Turbo	89.4	61.1	2.8	15.2	10.6	0.6	93.1	63.3	2.0	10.5	5.5	0.6
Debate(GPT-4)@3round	95.2	67.3	6.7	22.7	13.1	2.3	97.8	72.4	11.3	17.4	12.1	3.7
EcoNash(GPT-4)	**100**	**71.4**	**15.6**	**32.1**	**25.7**	**7.2**	**100**	**82.1**	**26.6**	**32.4**	**17.6**	**9.3**
Sole-planning												
Direct$_{\text{GPT-3.5-Turbo}}$	100	60.2	4.4	11.0	2.8	0	100	59.5	2.7	9.5	4.4	0.6
CoT$_{\text{GPT-3.5-Turbo}}$	100	66.3	3.3	11.9	5.0	0	100	64.4	2.3	9.8	3.8	0.4
ReAct$_{\text{GPT-3.5-Turbo}}$	82.2	47.6	3.9	11.4	6.7	0.6	81.6	45.9	2.5	10.7	3.1	0.7
Reflexion$_{\text{GPT-3.5-Turbo}}$	93.9	53.8	2.8	11.0	2.8	0	92.1	52.1	2.2	9.9	3.8	0.6
Direct$_{\text{Mixtral-8x7B-MoE}}$	100	68.1	5.0	3.3	1.1	0	99.3	67.0	3.7	3.9	1.6	0.7
Direct$_{\text{Gemini Pro}}$	93.9	65.0	8.3	9.3	4.4	0.6	93.7	64.7	7.9	10.6	4.7	2.1
Direct$_{\text{GPT-4-Turbo}}$	**100**	80.4	17.2	47.1	22.2	4.4	**100**	80.6	15.2	44.3	23.1	4.4
Debate(GPT-4)	97.7	78.9	15.6	43.3	20.6	6.7	98.2	79.5	18.8	41.7	22.9	7.1
EcoNash(GPT-4)	**100**	**83.3**	**22.2**	**51.7**	**27.8**	**12.9**	**100**	**84.2**	**23.5**	**49.8**	**28.7**	**15.2**

Table 3: Performance of different configurations in Execution LLMs on GSM-Hard and MATH.

Method	**GSM-Hard**	**MATH**
Baselines		
EcoNash	51.43	81.47
LLaMA 3.1 70b (Few-shot CoT)	42.23	62.71
EcoNash Configurations		
Homog. (3× Llama3.1 8b)	48.71	67.70
Homog. (3× Llama3.1 405b)	61.29	89.24
Heterog. (Llama3.1 8b, Llama3 8b, Mixtral 7b)	45.24	74.24
Heterog. (Mixtral 8×22b, Qwen1.5 110b, Llama3.1 405b)	55.73	85.46

Execution LLMs and one coordinator, the scaled-up system with 9 Execution LLMs, 3 coordinators, and a central LLM achieved 18.1% improvement in Figure3, which has potential to further scale up.

4.5 Ablation Study

In the additional experiments, heterogeneous Execution LLMs experienced a slight performance decline. An intuitive explanation for this observation is that implementing BNE is more challenging for heterogeneous Execution LLMs. To verify the actual performance differences of the EcoNash framework before and after achieving BNE, we conducted experiments on three math reasoning benchmarks: GSM8K, GSM-Hard, and MATH. Results in Table 5 demonstrate that our framework achieved an average performance improvement of 14% after implementing BNE.

Table 4: Average token usage and performance comaprison in the Math, GSM8K, and GSM-Hard.

Dataset	Inference Strategy	LLaMA3.1 70B		Mixtral 8x7b		Mixtral 8x22b	
		Token Usage	Performance	Token Usage	Performance	Token Usage	Performance
Math	EcoNash	1629.79	81.47	1128.23	35.02	4270.86	72.29
	Multi-Agent Debate (3 rounds)	2154.87	71.58	1462.12	31.28	5345.56	67.41
	RAP	2653.27	68.71	1737.73	33.99	6668.55	62.53
	EcoNash (with detailed strategy)	3270.06	72.38	2150.23	26.18	8054.03	68.23
	Self Consistency (64 rounds)	11917.00	67.39	8066.21	31.58	29616.13	62.21
GSM8K	EcoNash	1131.65	92.70	1284.98	76.97	4715.31	88.20
	Multi-Agent Debate (3 rounds)	1391.57	86.32	1463.40	70.19	5714.05	81.95
	RAP	1907.86	81.33	1248.66	72.03	6517.77	76.97
	EcoNash (with detailed strategy)	2772.24	85.17	1188.13	65.37	9341.60	81.46
	Self Consistency (64 rounds)	9574.25	89.56	6601.34	71.08	24671.91	86.24
GSM-Hard	EcoNash	1518.76	51.43	1271.53	25.76	7101.62	47.58
	Multi-Agent Debate (3 rounds)	3030.73	41.98	1478.14	20.04	9250.78	45.21
	RAP	1768.72	38.97	1036.11	22.47	6464.52	42.79
	EcoNash (with detailed strategy)	3662.64	44.12	2239.07	18.52	11464.98	41.04
	Self Consistency (64 rounds)	16724.69	39.76	11668.19	22.47	74544.25	44.19

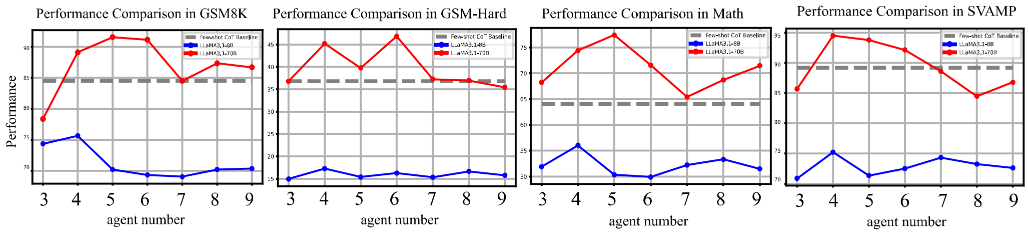

Figure 2: Scaling up our framework with a single coordinator while increasing the number of Execution LLMs. Experiments were conducted on GSM8K, GSM-Hard, Math, and SVAMP datasets.

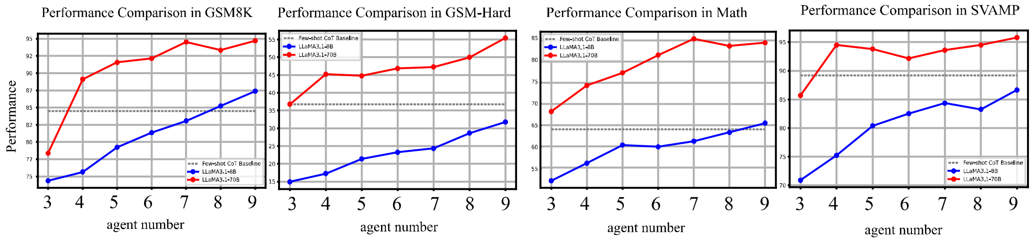

Figure 3: Scaling up our framework involves increasing the number of coordinators in proportion to the growing number of Execution LLMs, with coordinators scaling accordingly. Experiments were conducted on the GSM8K, GSM-Hard, MATH, and SVAMP datasets.

Table 5: Performance comparison of models with and without BNE across different datasets.

Dataset	Model	Without BNE (%)	With BNE (%)
GSM8K	LLaMA3.1-8B	74.38	80.33
	LLaMA3.1-70B	82.12	96.61
	LLaMA3.1-405B	92.36	100.00
GSM-Hard	LLaMA3.1-8B	21.73	30.71
	LLaMA3.1-70B	43.58	60.26
	LLaMA3.1-405B	51.54	65.91
MATH	LLaMA3.1-8B	55.92	71.45
	LLaMA3.1-70B	74.47	87.31
	LLaMA3.1-405B	82.31	94.78

Table 6: Ablation on reward.

R_1	R_2	R_3	EcoNash
✓	×	✓	77.55
✓	×	×	74.32
✓	✓	×	76.21
Random			62.71

Table 7: Ablation on strategy.

S_1	S_2	S_3	EcoNash
✓	×	×	71.35
×	✓	×	72.31
×	×	✓	81.47

Additionally, we performed ablation studies on various submodules, including the reward design and the setting where the Coordinator LLM provides a strategy without giving a direct answer, to ensure the validity of our architecture. All experiments were conducted with `Llama 3.1 70B model`, tested on the MATH benchmark. Specifically, R_1 refers to the action likelihood reward, R_2 to the task-specific reward, and R_3 to the self-evaluation reward. S_1 represents the setting where the coordinator does not provide any strategy, while S_2 represents the setting where the coordinator provides both a detail strategy, S_3 represents EcoNash, with informative strategy as our baseline.

5 Conclusion

In this work, we introduce EcoNash, a novel collaborative reasoning framework in multi-LLM systems. EcoNash constructs a hierarchical coordination mechanism, enabling multiple Execution LLMs to engage in distributed reasoning guided by a Coordinator LLM. The hierarchical coordination mechanism allows each Execution LLM to operate independently with its own belief network, receiving only the question and strategy from the Coordinator LLM. This enables multiple Execution LLMs to engage in distributed reasoning, guided by the Coordinator LLM, to achieve BNE. Experimental results across six benchmarks demonstrate EcoNash outperforms single-agent approaches by 10.9% and surpasses the performance of existing multi-agent methods by 11.2% in average, confirming the robustness and efficiency of our framework. Moreover, EcoNash demonstrate great potential to scale up the mulit-LLMs system while maintain relatively reasonable consumption cost.

Ethic Statement

The study does not involve human subjects, data set releases, potentially harmful insights, applications, conflicts of interest, sponsorship, discrimination, bias, fairness concerns, privacy or security issues, legal compliance issues, or research integrity issues.

Reproducibility Statement

The experimental setups for training and evaluation are described in detail in Section 4.1, and the experiments are all conducted using public datasets. We provide the link to our source codes to ensure the reproducibility of our experimental results: `https://anonymous.4open.science/status/EcoNash-867A`.

References

Jamshaid Ahmad, Abdullah Eqal Al-Mazrooei, and Themistocles M Rassias. Common fixed point theorems with applications to theoretical computer science. *International Journal of Nonlinear Analysis and Applications*, 14(2):1–10, 2023.

Vivek S Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, Cambridge, UK, 2009.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. In *ICLR*, 2024.

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

Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*, 2023.

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

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *ICML*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning to model other minds: A deep learning framework for social intelligence. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, pp. 8112–8122, 2018a.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018b.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

Drew Fudenberg and David K Levine. *The Theory of Learning in Games*. MIT Press, Cambridge, MA, 1998.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *ICML*, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Elad Hazan. *Introduction to Online Convex Optimization*. Now Publishers Inc, Boston, MA, 2016.

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

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3040–3049, 2019.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on learning theory*, pp. 2137–2143. PMLR, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *NeurIPS*, 2022.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Xiaochong Lan, Chen Gao, Depeng Jin, and Yong Li. Stance detection with collaborative role-infused llm-based agents. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 18, pp. 891–903, 2024.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4190–4203, 2017.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. In *First Conference on Language Modeling*, 2024.

Jie Liu, Zhiwei Ding, Yong Liu, and Xinwei Wang. Decentralized multi-agent reinforcement learning with networked agents: Recent advances. *Foundations and Trends in Machine Learning*, 15 (1):1–120, 2022.

Li-Chun Lu, Shou-Jen Chen, Tsung-Min Pai, Chan-Hung Yu, Hung-yi Lee, and Shao-Hua Sun. Llm discussion: Enhancing the creativity of large language models via discussion framework and role-play. *arXiv preprint arXiv:2405.06373*, 2024.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

Martin Owe and Christopher A Sims. Information theoretic limits of strategic communication. *Journal of Economic Theory*, 148(6):2404–2434, 2013.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. *arXiv preprint arXiv:2310.06272*, 2023.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*, 2024.

Shai Shalev-Shwartz. *Online Learning and Online Convex Optimization*. Now Publishers Inc, Boston, MA, 2012.

Andries Petrus Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D Barrett, and Arnu Pretorius. Should we be going mad? a look at multi-agent debate strategies for llms. In *ICML*, 2024.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018.

Xingchen Wan, Ruoxi Sun, Hanjun Dai, Sercan O Arik, and Tomas Pfister. Better zero-shot reasoning with self-adaptive prompting. *arXiv preprint arXiv:2305.14106*, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*, 2023.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*, 2024a.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36, 2024b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuan-Jing Huang, and Xipeng Qiu. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15135–15153, 2023.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. *arXiv preprint arXiv:2406.14228*, 2024a.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. Following length constraints in instructions. *arXiv preprint arXiv:2406.17744*, 2024b.

Jintian Zhang, Xin Xu, and Shumin Deng. Exploring collaboration mechanisms for llm agents: A social psychology view. *arXiv preprint arXiv:2310.02124*, 2023a.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent deep reinforcement learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(6):503–527, 2021.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *ICLR*, 2023b.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Banghua Zhu, Michael Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In *International Conference on Machine Learning*, pp. 43037–43067. PMLR, 2023.

A THEORETICAL PROOF

A.1 PROOF OF THEOREM 1

Proof. We aim to prove the existence of a Bayesian Nash Equilibrium (BNE) in our multi-agent LLM framework under the specified conditions. The proof proceeds by verifying the conditions of Glicksberg's Fixed Point Theorem, which guarantees the existence of a fixed point in continuous games with infinite-dimensional strategy spaces.

Step 1: Define the Best Response Correspondence

For each agent i, define the best response correspondence BR_i as:

$$BR_i(\pi_{-i}) = \{\pi_i \in \Pi_i \mid \pi_i \text{ maximizes } U_i\left(\theta_i, \pi_i, \pi_{-i}\right)\},$$

where Π_i is the set of all admissible strategies for agent i, and π_{-i} denotes the strategies of all other agents.

Step 2: Verify the Conditions of Glicksberg's Fixed Point Theorem

To apply Glicksberg's Fixed Point Theorem, we need to verify the following conditions for each agent i:

1. *Strategy Space Compactness and Convexity*:
   - The strategy space Π_i is non-empty, convex, and compact in the topology of pointwise convergence.
2. *Continuity of Payoff Functions*:
   - The payoff function $U_i(\theta_i, \pi_i, \pi_{-i})$ is continuous in (π_i, π_{-i}) for each fixed θ_i.
3. *Quasi-Concavity of Payoff Functions*:
   - The payoff function $U_i(\theta_i, \pi_i, \pi_{-i})$ is quasi-concave in π_i for each fixed θ_i and π_{-i}.

Verification:

1. **Strategy Space Compactness and Convexity**:

   The strategy space Π_i consists of all measurable functions mapping types θ_i to actions a_i in $\mathcal{A}_i$. Since Θ_i and $\mathcal{A}_i$ are compact metric spaces, and strategies are measurable functions from one compact space to another, the space of such functions Π_i can be endowed with the topology of pointwise convergence, making it compact by Tychonoff's Theorem. Convexity follows because the set of mixed (probabilistic) strategies is convex, and any convex combination of measurable functions is measurable.

2. **Continuity of Payoff Functions**:

   For fixed θ_i, the payoff function $U_i(\theta_i, \pi_i, \pi_{-i})$ depends continuously on π_i and π_{-i} due to the continuity of U_i in actions and types. Specifically, since U_i is continuous in $a = (a_i, a_{-i})$ and the strategies π_i, π_{-i} map continuously from types to actions, the composition $U_i(\theta_i, \pi_i(\theta_i), \pi_{-i}(\theta_{-i}))$ is continuous in (π_i, π_{-i}).

3. **Quasi-Concavity of Payoff Functions**:

   For each θ_i and π_{-i}, the function $\pi_i \mapsto U_i(\theta_i, \pi_i, \pi_{-i})$ is quasi-concave because U_i is quasi-concave in a_i and the strategies are linear in the space of mixed strategies. Therefore, any convex combination of strategies does not decrease the utility, satisfying quasi-concavity.

Step 3: Establish Upper Hemicontinuity and Non-Empty, Convex-Valuedness of Best Response Correspondences

We need to show that $BR_i(\pi_{-i})$ is upper hemicontinuous with non-empty, convex values.

1. **Non-Empty, Convex Values**:

   For each π_{-i}, since Π_i is compact and convex, and U_i is continuous and quasi-concave in π_i, the Weierstrass Theorem ensures that the maximum exists; hence, $BR_i(\pi_{-i})$ is non-empty. Convexity follows from the quasi-concavity of U_i in π_i, implying that any convex combination of best responses is also a best response.

2. **Upper Hemicontinuity**:

   Upper hemicontinuity of BR_i means that for any net $\pi_{-i}^{\alpha} \to \pi_{-i}$, and any $\pi_i \in BR_i(\pi_{-i})$, there exists a net $\pi_i^{\alpha} \in BR_i(\pi_{-i}^{\alpha})$ such that $\pi_i^{\alpha} \to \pi_i$. This property holds because the payoff function U_i is continuous in (π_i, π_{-i}), and the strategy spaces are compact.

Step 4: Application of Glicksberg's Fixed Point Theorem

Having verified all the conditions, we can apply Glicksberg's Fixed Point Theorem, which states that if each player's strategy set is compact and convex, and their payoff functions are continuous and quasi-concave in their own strategies, then the game has at least one Nash Equilibrium in mixed strategies.

Step 5: Conclusion

Therefore, there exists a strategy profile $\overline{\pi}^* = (\pi_1^*, \pi_2^*, \ldots, \pi_N^*)$ such that for each agent i,

$$\pi_i^* \in BR_i(\pi_{-i}^*),$$

meaning that no agent can unilaterally deviate to improve their expected payoff, given their beliefs about other agents' types and strategies. This strategy profile constitutes a Bayesian Nash Equilibrium in our multi-agent LLM framework.

□

A.2 Proof of Proposition 1

Proof. We aim to show that, by minimizing the TD loss for each agent's Q-network, the agents' strategies converge to a Bayesian Nash Equilibrium (BNE).

Assumptions:

1. The Q-networks $Q_i(\mathbf{s}, a_i; \theta_i)$ are parameterized by prompt embeddings θ_i, and the mapping from θ_i to Q_i is continuously differentiable.

2. The exploration strategy ensures sufficient coverage of the state-action space (e.g., ϵ-greedy with decaying ϵ).

3. The loss function $L_i(\theta_i)$ is convex or has Lipschitz continuous gradients with respect to θ_i.

4. The gradient $\nabla_{\theta_i} L_i(\theta_i)$ is Lipschitz continuous.

5. The learning rate η_t is chosen such that it satisfies the Robbins-Monro conditions: $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$.

Step 1: Defining the TD Loss Function The TD loss function for agent i is:

$$L_i(\theta_i) = \mathbb{E}_{(\mathbf{s}, a_i, r_i, \mathbf{s}') \sim \mathcal{D}_i} \left[\left(r_i + \gamma \max_{a_i'} Q_i(\mathbf{s}', a_i'; \theta_i^-) - Q_i(\mathbf{s}, a_i; \theta_i) \right)^2 \right]$$

This loss measures the discrepancy between the predicted Q-value and the target Q-value based on the reward and the estimated optimal future Q-value.

Step 2: Gradient Descent Update Agent i updates its Q-network parameters according to:

$$\theta_i^{t+1} = \theta_i^t - \eta_t \cdot \nabla_{\theta_i} L_i(\theta_i^t).$$

The gradient of the loss function with respect to the parameters is:

$$\nabla_{\theta_i} L_i(\theta_i^t) = \mathbb{E}_{(\mathbf{s},a_i,r_i,\mathbf{s}')\sim\mathcal{D}_i}\left[2\left(r_i + \gamma \max_{a_i'} Q_i(\mathbf{s}', a_i'; \theta_i^-) - Q_i(\mathbf{s}, a_i; \theta_i^t)\right)\cdot(-\nabla_{\theta_i} Q_i(\mathbf{s}, a_i; \theta_i^t))\right].$$

Step 3: Convergence of Gradient Descent with TD Loss Under the assumptions that $L_i(\theta_i)$ has Lipschitz continuous gradients and the learning rate η_t satisfies the Robbins-Monro conditions, stochastic gradient descent converges to a stationary point θ_i^* of $L_i(\theta_i)$:

$$\lim_{t\to\infty} \theta_i^t = \theta_i^*.$$

At convergence, the gradient vanishes:

$$\nabla_{\theta_i} L_i(\theta_i^*) = 0,$$

which implies:

$$\mathbb{E}_{(\mathbf{s},a_i,r_i,\mathbf{s}')\sim\mathcal{D}_i}\left[\left(r_i + \gamma \max_{a_i'} Q_i(\mathbf{s}', a_i'; \theta_i^-) - Q_i(\mathbf{s}, a_i; \theta_i^*)\right)\cdot\nabla_{\theta_i} Q_i(\mathbf{s}, a_i; \theta_i^*)\right] = 0.$$

Assuming that the Q-network parameterization is such that the above condition holds only when:

$$Q_i(\mathbf{s}, a_i; \theta_i^*) = r_i + \gamma \max_{a_i'} Q_i(\mathbf{s}', a_i'; \theta_i^-),$$

the Q-network accurately estimates the expected cumulative rewards, aligning the agent's policy with the optimal response to other agents' strategies.

Step 4: Characterizing the Stationary Point At the stationary point θ_i^*, the Q-network satisfies the Bellman optimality condition:

$$Q_i(\mathbf{s}, a_i; \theta_i^*) = r_i + \gamma \max_{a_i'} Q_i(\mathbf{s}', a_i'; \theta_i^-).$$

This condition ensures that the agent's policy $\pi_i(a_i \mid \mathbf{s}; \theta_i^*)$ is a best response to the current policies of other agents, as it maximizes the expected cumulative reward.

Step 5: Establishing Bayesian Nash Equilibrium Since each agent's policy is a best response to the policies of others, the set of policies $\{\pi_i^*\}$ constitutes a Bayesian Nash Equilibrium. Each agent maximizes its expected utility given its beliefs about other agents' types and strategies, fulfilling the definition of BNE.

□

A.3 Assumptions

Our theoretical analysis relies on four key assumptions that are both common in multi-agent systems Zhang et al. (2021); Liu et al. (2022) and specifically relevant to our MA-LLM framework.

Definition 1 (System Components). *In our MA-LLM framework:*

- *Each agent i's observation $O_i = [e_t, e_s, \mathbf{b}_i]^\top$, where e_t encodes the task, e_s represents the coordinator's strategy, and $\mathbf{b}_i$ is the belief state*
- *Each agent's action is its prompt embedding $\mathbf{e}_i$ generated by belief network $B_i(\tau_i, O_i; \theta_i^B)$*
- *The coordinator aggregates beliefs through $f_e(\{\mathbf{b}_i\}_{i=1}^N; \theta_e)$ into group information $\mathbf{E}$*

Assumption 1 (Bounded Rewards). *The rewards from coordinator commitment are uniformly bounded: $|r_i(O_i, \mathbf{e}_i, \mathbf{E})| \leq R_{\max}$, for all $O_i, \mathbf{e}_i, \mathbf{E}, i$.*

This assumption is standard in reinforcement learning Sutton & Barto (2018) and critical since it ensures numerical stability in the learning process of LLMs, preventing reward explosion that could lead to unstable training.

Definition 2 (Historical Data and Posterior). *Given historical data $D_t = \{(O_i^k, \mathbf{e}_i^k, C^k)\}_{k=1}^t$:*

- $P_{post}(\mathbf{E} \mid D_t, O_i, \mathbf{e}_i)$ *is the posterior distribution over group information determined by the coordinator*
- $P_{LLM}(\mathbf{E} \mid D_t, O_i, \mathbf{e}_i)$ *is the belief distribution maintained by each execution LLM*

Assumption 2 (Approximate Posterior Alignment). *Execution LLMs aim to align with the posterior distributions determined by the Coordinator LLM within an acceptable error margin* $\epsilon > 0$*:*

$$D_{KL}\left(P_{LLM}(\mathbf{E} \mid D_t, O_i, \mathbf{e}_i) \,\|\, P_{post}(\mathbf{E} \mid D_t, O_i, \mathbf{e}_i)\right) \leq \epsilon,$$

where D_{KL} *denotes the Kullback-Leibler divergence.*

This approximate alignment acknowledges that perfect alignment is impractical but strives for a close approximation:

- The Coordinator LLM acts as a centralized distributor of strategic guidance.
- Execution LLMs maintain belief alignment through prompt (detailed in Section 3.3.2).
- Monotonic guarantee in EcoNash mixing optimization network A.5.
- Such alignment has been shown in Foerster et al. (2018a); Jaques et al. (2019) to enhance coordination.

Definition 3 (Belief Entropy). *For a given time t, the belief entropy* H_t *is defined as the Shannon entropy of the aggregated belief embeddings:*

$$H_t = -\sum_{i=1}^{N} \mathbb{E}_{\mathbf{b}_i \sim B_i}[\mathbf{b}_i \log \mathbf{b}_i]$$

where B_i *represents the belief network of agent i.*

Assumption 3 (Game Regularity). *There exists* $\eta > 0$ *such that for any* $t_1 < t_2$*, if* $H_{t_1} - H_{t_2} \leq \log 2$*, then*

$$I(\theta_i^B; \xi(\mathbf{e}_i, \mathbf{E}) \mid D_{t_1}) \leq 4\eta \cdot I(\theta_i^B; \xi(\mathbf{e}_i, \mathbf{E}) \mid D_{t_2}),$$

for all agents i*, where* θ_i^B *are the belief network parameters.*

This information-theoretic assumption serves multiple purposes in our framework:

- It ensures the stability of belief updates between LLMs over time by bounding the entropy difference of belief states.
- The mutual information term $I(\theta_i^B; \xi(\mathbf{e}_i, \mathbf{E}))$ quantifies how much an LLM's belief network parameters affect its coordination through prompt embeddings.
- The bound 4η controls the rate at which LLMs can adapt their belief states based on observed interactions and coordinator guidance.

Definition 4 (Value Function and Bellman Operator). *For each execution LLM i:*

- *The value function* $V_t(O_i) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | O_i^t = O_i]$ *estimates the expected cumulative rewards*
- *The optimal prompt embeddings* $\mathbf{e}_i^{*t}$ *maximize the Q-function* $Q_i(O_i, \mathbf{e}_i; \theta_i^B)$ *at time t*
- *The Bellman operator* B_t *transforms one value function to another:* $(B_t V)(O_i) = \max_{\mathbf{e}_i} \mathbb{E}[r_i + \gamma V(O_i') | O_i, \mathbf{e}_i]$

Assumption 4 (Concentrability). *There exists* $\kappa < \infty$ *such that*

$$\mathbb{E}\left[\sum_{t=1}^{T} \sum_{i=1}^{N} \left((B_t - B^*)V_t\right)^2 (O_i^t, \mathbf{e}_i^{*t}, \mathbf{E}^{*t})\right] \leq \kappa^2 T,$$

where B^* *is the true Bellman operator.*

This assumption is fundamental to our theoretical guarantees:

- It ensures that the value function estimates by each LLM converge to their true values at an appropriate rate.
- The constant κ bounds the cumulative estimation error across all LLMs, critical for establishing our regret bounds.
- In our MA-LLM system, this translates to the stability of response quality improvements during training.

Collective Impact: Together, these assumptions enable us to:

- Establish the existence of BNE in our MA-LLM system (Theorem 1)
- Derive meaningful regret bounds for the learning process (Lemma 1)
- Guarantee the convergence of our iterative training procedure (Proposition 1)

A.4 Scaling Up the System

To extend our framework to larger systems, we implement a hierarchical structure where clusters of Coordinator LLMs and their associated Execution LLMs form local Nash Equilibria, which are then coordinated through a global Coordinator LLM to establish a global Nash Equilibrium. This hierarchical design preserves our theoretical guarantees while enabling efficient scaling. The process is detailed in Algorithm 2.

Algorithm 2 Scaling-Up Framework for EcoNash

Require: Global Coordinator LLM Coordglobal, Local Coordinator LLMs Coord$kk = 1^K$
Require: System parameters ϵ_C, Rthreshold, ϵ_L, Learning rates $\eta, \eta', \eta_{\text{global}}$
Ensure: Optimized hierarchical Nash Equilibrium

Initialize cluster embeddings $\mathbf{E}kk = 1^K$ and prompt embeddings $\mathbf{e}i$ for all LLMs
while not converged **do**
  $\mathbf{S} \leftarrow$ Coordglobal(e_t) ▷ Global strategy generation
  for each cluster $k = 1$ to K **in parallel do**
    $O_k \leftarrow [e_t, \mathbf{S}, \mathbf{E}_k]^\top$ ▷ Cluster observation
    Local strategy: $\mathbf{s}k \leftarrow$ Coord$k(O_k)$
    for each Execution LLM $i \in C_k$ **in parallel do**
      $O_i \leftarrow [e_t, \mathbf{s}k, \mathbf{b}i]^\top$ ▷ Agent observation
      Generate output u_i with parameters (T_i, p_i)
      Compute rewards:
        $r_i^{\text{AL}} \leftarrow \min(R\text{max}, \text{sim}(u_i, c_k))$
        $r_i^{\text{TS}} \leftarrow \min(R\text{max}, \text{eval}(u_i, \text{task}))$
        $r_i^{\text{CC}} \leftarrow \min(R\text{max}, \text{quality}(u_i, u_j j \in C_k))$
      $r_i \leftarrow \alpha_1 r_i^{\text{AL}} + \alpha_2 r_i^{\text{TS}} + \alpha_3 r_i^{\text{CC}}$
      Update belief network using loss $L_i(\theta_i^B)$
    end for
    $c_k \leftarrow$ Coord$k(u_i i \in C_k)$ ▷ Local commitment
    Update cluster embedding $\mathbf{E}_k$ using local metrics
  end for Copy
  $C \leftarrow \text{Coord}_{\text{global}}(\{c_k\}_{k=1}^K)$ ▷ Global commitment
  for each cluster $k = 1$ to K **do**
    Compute global reward: $R_k \leftarrow R_{\text{global}}(\text{sim}(c_k, C))$
    Update local Coordinator parameters
  end for
  Early Stopping Check:
  if $\|C_{t+1} - C_t\| \leq \epsilon_C$ **and** $\frac{1}{K}\sum_{k=1}^K R_k \geq R_{\text{threshold}}$ **then**
    break
  end if
end while

A.4.1 DETAILED EXPLANATION

Initialization

- **Clustering**: Execution LLMs are divided into K clusters $\{C_1, C_2, \ldots, C_K\}$ based on task similarity.
- **Local Coordinator LLMs**: Each cluster C_k is assigned a local Coordinator LLM Coord$_k$ to manage its Execution LLMs.
- **Global Coordinator LLM**: A Central LLM $Central$ oversees all clusters.
- **Embeddings**: Initialize prompt embeddings $\mathbf{e}_i$ for Execution LLMs and cluster embeddings $\mathbf{E}_k$ for clusters.

Global Strategy Generation The global Coordinator LLM generates a high-level strategy $\mathbf{S}$ based on the question q. This strategy provides overall guidance and is distributed to all local Coordinator LLMs.

Local Inference and Optimization Each local Coordinator LLM Coord$_k$ generates a local strategy $\mathbf{s}_k$ using $\mathbf{S}$ and the cluster embedding $\mathbf{E}_k$. Execution LLMs within the cluster receive $(q, \mathbf{s}_k, \mathbf{e}_i)$ and generate individual answers a_i. The local Coordinator LLM aggregates these answers to form a local commitment c_k.

Local Optimization Execution LLMs compute local rewards based on the similarity between their answers and the local commitment. Prompt embeddings $\mathbf{e}_i$ are updated to maximize expected rewards. Cluster embeddings $\mathbf{E}_k$ are also updated to improve Coordinator at the cluster level.

Global Commitment Formation The global Coordinator LLM aggregates local commitments $\{c_k\}$ to form the final global commitment C, representing the system's overall response.

Global Optimization Each cluster receives a global reward R_k based on the similarity between its local commitment c_k and the global commitment C. Local Coordinator LLMs are updated based on the global rewards to improve alignment with the global objective.

Convergence Check The system checks if global convergence criteria are met, such as minimal changes in the global commitment or reaching a performance threshold. If met, the algorithm terminates; otherwise, it proceeds to the next episode.

A.5 PROOF OF MIXING NETWORK MONOTONICITY

Proposition 2 (Monotonicity of Mixing Network)**.** *The mixing network Q_{tot} is monotonic in each individual Q-value Q_i, ensuring that improvements in Q_i lead to improvements in Q_{tot}.*

Proof. The mixing network is designed using positive weights and non-decreasing activation functions. Specifically, let the mixing network be composed of layers where each layer l computes:

$$h^l = \phi^l(W^l h^{l-1} + b^l)$$

where:

- $h^0 = [Q_1, Q_2, \ldots, Q_N]^\top$
- W^l has non-negative entries.
- ϕ^l is a non-decreasing activation function (e.g., ReLU).

We proceed by induction to show that each component of h^l is a non-decreasing function of Q_i.

Base Case: At layer $l = 0$, $h_i^0 = Q_i$, so $\frac{\partial h_i^0}{\partial Q_j} = \delta_{ij} \geq 0$.

Inductive Step: Assume $\frac{\partial h_k^{l-1}}{\partial Q_i} \geq 0$ for all k. Then, for each component h_j^l:

$$h_j^l = \phi^l \left(\sum_k W_{jk}^l h_k^{l-1} + b_j^l \right)$$

Since $W_{jk}^l \geq 0$ and ϕ^l is non-decreasing:

$$\frac{\partial h_j^l}{\partial Q_i} = \phi'^l \left(\sum_k W_{jk}^l h_k^{l-1} + b_j^l \right) \sum_k W_{jk}^l \frac{\partial h_k^{l-1}}{\partial Q_i} \geq 0$$

because $\phi'^l \geq 0$ and $\frac{\partial h_k^{l-1}}{\partial Q_i} \geq 0$ by the inductive hypothesis. Therefore, $\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0$, ensuring monotonicity. □

This monotonicity property is crucial as it ensures that improvements in individual agent performances contribute positively to the overall system performance, aligning local and global objectives within EcoNash.

B Detailed Proofs

B.1 Proof of Lemma 1

Proof. Consider the value functions under policies π' and π:

$$V_i^{\pi'}(s) = \mathbb{E}_{\pi'} \left[\sum_{k=0}^{\infty} \gamma^k r_i(s_k, a_k) \mid s_0 = s \right], \quad V_i^{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_i(s_k, a_k) \mid s_0 = s \right].$$

Their difference is:

$$\begin{aligned} V_i^{\pi'}(s) - V_i^{\pi}(s) &= \mathbb{E}_{\pi'} \left[\sum_{k=0}^{\infty} \gamma^k r_i(s_k, a_k) \right] - \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_i(s_k, a_k) \right] \\ &= \sum_{k=0}^{\infty} \gamma^k \left(\mathbb{E}_{s_k \sim d_{\pi'}^k} \left[r_i(s_k, a_k) \right] - \mathbb{E}_{s_k \sim d_{\pi}^k} \left[r_i(s_k, a_k) \right] \right). \end{aligned}$$

Assuming the difference in state distributions is negligible (justified under Assumption 4), we focus on action differences. Using the Q-function definition:

$$Q_i^{\pi}(s, a_i, a_{-i}) = r_i(s, a_i, a_{-i}) + \gamma \mathbb{E}_{s' \sim P} \left[V_i^{\pi}(s') \right],$$

we can write:

$$V_i^{\pi'}(s) - V_i^{\pi}(s) = \sum_{k=0}^{\infty} \gamma^k \mathbb{E}_{s_k \sim d_{\pi'}^k} \left[Q_i^{\pi}(s_k, a_k') - V_i^{\pi}(s_k) \right].$$

Since $V_i^{\pi}(s_k) = \mathbb{E}_{a_k \sim \pi(s_k)} \left[Q_i^{\pi}(s_k, a_k) \right]$, we have:

$$V_i^{\pi'}(s) - V_i^{\pi}(s) = \sum_{k=0}^{\infty} \gamma^k \mathbb{E}_{s_k \sim d_{\pi'}^k} \left[\mathbb{E}_{a_k' \sim \pi'(s_k)} \left[Q_i^{\pi}(s_k, a_k') - \mathbb{E}_{a_k \sim \pi(s_k)} \left[Q_i^{\pi}(s_k, a_k) \right] \right] \right].$$

Switching the order of expectations and summing over k, we get:

$$V_i^{\pi'}(s) - V_i^{\pi}(s) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{\pi'}} \left[Q_i^{\pi}(s, a_i', a_{-i}') - Q_i^{\pi}(s, a_i, a_{-i}) \right].$$

□

B.2 Bounding the Bayesian Regret

Starting from the regret definition for agent i over T steps:

$$R_i(T) = \mathbb{E}_{s_t,\pi_t}\left[\sum_{t=1}^{T}\left(V_i^*(s_t) - V_i^{\pi_t}(s_t)\right)\right],$$

where the expectation is over the randomness in state transitions and policies.

Applying Lemma 1:

$$V_i^*(s_t) - V_i^{\pi_t}(s_t) = \frac{1}{1-\gamma}\mathbb{E}_{a_i^{*t},a_{-i}^{*t},a_i^t,a_{-i}^t}\left[Q_i^{\pi_t}(s_t,a_i^{*t},a_{-i}^{*t}) - Q_i^{\pi_t}(s_t,a_i^t,a_{-i}^t)\right].$$

We decompose the Q-value difference:

$$\begin{aligned}
&Q_i^{\pi_t}(s_t,a_i^{*t},a_{-i}^{*t}) - Q_i^{\pi_t}(s_t,a_i^t,a_{-i}^t)\\
&= \left(Q_i^{\pi_t}(s_t,a_i^{*t},a_{-i}^{*t}) - Q_i^{*}(s_t,a_i^{*t},a_{-i}^{*t})\right) && \text{(Error Term 1)}\\
&\quad + \left(Q_i^{*}(s_t,a_i^{*t},a_{-i}^{*t}) - Q_i^{*}(s_t,a_i^{t},a_{-i}^{t})\right) && \text{(Policy Suboptimality)}\\
&\quad + \left(Q_i^{*}(s_t,a_i^{t},a_{-i}^{t}) - Q_i^{\pi_t}(s_t,a_i^{t},a_{-i}^{t})\right). && \text{(Error Term 2)}
\end{aligned}$$

Define the Q-function estimation error:

$$\epsilon_t = \max_{s,a_i,a_{-i}}\left|Q_i^{\pi_t}(s,a_i,a_{-i}) - Q_i^*(s,a_i,a_{-i})\right|.$$

Assumption 5 (Q-function Estimation Error)**.** *The estimation error decreases as:*

$$\epsilon_t \le \frac{C_\epsilon}{t^\alpha}, \quad \textit{with } \alpha = \frac{1}{2}.$$

This rate is justified by:

- *Stochastic approximation theory showing $O(t^{-1/2})$ convergence (Borkar (2009)).*
- *Minimax optimality in stochastic optimization (Nemirovski et al. (2009)).*
- *Achievement through proper learning rate scheduling.*

Assumption 6 (Policy Suboptimality)**.** *The policy suboptimality decreases as:*

$$\delta_t \le \frac{C_\delta}{t^\beta}, \quad \textit{with } \beta = \frac{1}{2}.$$

This rate is supported by:

- *Regret bounds in online learning (Hazan (2016)).*
- *Gradient-based methods in convex policy spaces (Shalev-Shwartz (2012)).*
- *Empirical evidence in cooperative multi-agent RL (Zhang et al. (2021)).*

Using these assumptions, we have:

$$Q_i^{\pi_t}(s_t,a_i^{*t},a_{-i}^{*t}) - Q_i^{\pi_t}(s_t,a_i^t,a_{-i}^t) \le 2\epsilon_t + \delta_t.$$

Summing over t and all agents:

$$\begin{aligned}
R(T) &\le \sum_{i=1}^{N}\frac{1}{1-\gamma}\sum_{t=1}^{T}(2\epsilon_t + \delta_t)\\
&\le \sum_{i=1}^{N}\frac{1}{1-\gamma}\left(2C_\epsilon\sum_{t=1}^{T}\frac{1}{t^\alpha} + C_\delta\sum_{t=1}^{T}\frac{1}{t^\beta}\right)\\
&= O\left(\frac{N\sqrt{T}}{1-\gamma}\right).
\end{aligned}$$

B.3 Comparison with Multi-Agent Debate

In multi-agent debate settings, we analyze the regret bound using the same decomposition from Lemma 1:

Assumption 7 (Persistent Policy Suboptimality in Debate)**.**

$$\delta_t \geq \delta_{\min} > 0$$

Justified by:

- *Game-theoretic properties of competitive settings Fudenberg & Levine (1998)*
- *Information-theoretic limitations Owe & Sims (2013)*
- *Empirical evidence of non-convergence Lanctot et al. (2017)*

Following the same decomposition from earlier:

$$V_i^*(s_t) - V_i^{\pi_t}(s_t) = \frac{1}{1-\gamma}\mathbb{E}_{a_i,a_{-i}}\left[Q_i^{\pi_t}(s_t, a_i^{*t}, a_{-i}^{*t}) - Q_i^{\pi_t}(s_t, a_i^t, a_{-i}^t)\right]$$

The Q-value difference still decomposes into three terms:

$$\begin{aligned} &Q_i^{\pi_t}(s_t, a_i^{*t}, a_{-i}^{*t}) - Q_i^{\pi_t}(s_t, a_i^t, a_{-i}^t) \\ &= \underbrace{\left(Q_i^{\pi_t}(s_t, a_i^{*t}, a_{-i}^{*t}) - Q_i^*(s_t, a_i^{*t}, a_{-i}^{*t})\right)}_{\leq \epsilon_t} \\ &\quad + \underbrace{\left(Q_i^*(s_t, a_i^{*t}, a_{-i}^{*t}) - Q_i^*(s_t, a_i^t, a_{-i}^t)\right)}_{\geq \delta_{\min}} \\ &\quad + \underbrace{\left(Q_i^*(s_t, a_i^t, a_{-i}^t) - Q_i^{\pi_t}(s_t, a_i^t, a_{-i}^t)\right)}_{\leq \epsilon_t} \end{aligned}$$

In the debate setting:

- The estimation error terms are still bounded by $\epsilon_t = \frac{C_\epsilon}{\sqrt{t}}$
- The policy suboptimality term is lower bounded by $\delta_{\min}$ (Assumption 7)

Therefore, for each agent i:

$$\begin{aligned} R_i(T) &= \mathbb{E}\left[\sum_{t=1}^{T}(V_i^*(s_t) - V_i^{\pi_t}(s_t))\right] \\ &\leq \frac{1}{1-\gamma}\sum_{t=1}^{T}(2\epsilon_t + \delta_{\min}) \\ &= \frac{1}{1-\gamma}\left(2C_\epsilon\sum_{t=1}^{T}\frac{1}{\sqrt{t}} + \delta_{\min}T\right) \\ &\leq \frac{1}{1-\gamma}\left(2C_\epsilon \cdot 2(\sqrt{T}-1) + \delta_{\min}T\right) \end{aligned}$$

Summing over all agents and noting that the $\delta_{\min}T$ term dominates:

$$R_{\text{debate}}(T) = O\left(\frac{N\delta_{\min}T}{1-\gamma}\right)$$

This linear growth contrasts with our framework's sublinear $O(N\sqrt{T})$ bound, demonstrating EcoNash's superior efficiency through coordinated learning toward BNE.

B.4 Detailed Reward Setting

The reward function R provides feedback on each agent's performance while respecting Assumption 1, ensuring all reward components are uniformly bounded by $R_{\max}$. Drawing inspiration from maximum entropy inverse reinforcement learning (Zhu et al., 2023), we define the Action Likelihood Reward $r_i^{\text{AL}} = \min(R_{\max}, \text{sim}(u_i, C))$, where $\text{sim}(u_i, C) = \frac{u_i \cdot C}{\|u_i\| \|C\|}$ measures the consistency between an agent's output u_i and the coordinator's commitment C. Following Hao et al. (2023), the Task-Specific Reward $r_i^{\text{TS}} = \min(R_{\max}, \text{eval}(u_i, \text{task}))$ evaluates domain-specific objectives through the coordinator's assessment, where eval computes normalized scores considering solution correctness in mathematical problems or response relevance in planning tasks. Building upon Xie et al. (2024b), the Collaborative Contribution Reward $r_i^{\text{CC}} = \min(R_{\max}, \text{quality}(u_i, \{u_j\}_{j \neq i}))$ enables the coordinator to assess each agent's output quality within the multi-agent context, where quality evaluates the response's coherence and creativity while considering its contribution to the collective solution. The total reward combines these components as $r_i = \alpha_1 r_i^{\text{AL}} + \alpha_2 r_i^{\text{TS}} + \alpha_3 r_i^{\text{CC}}$, where the weights $\alpha_1 + \alpha_2 + \alpha_3 = 1$ ensure the total reward is bounded by $R_{\max}$. To enhance adaptability and learning efficiency, we introduce a dynamic mechanism to adjusts these weights using gradient-based updates $\alpha_k \leftarrow \alpha_k - \eta_\alpha \cdot \partial \mathcal{L}_{\text{dr}} / \partial \alpha_k$, where $\mathcal{L}_{\text{dr}} = \sum_{i=1}^{N} (r_i^{\text{actual}} - r_i^{\text{expected}})^2$ measures the discrepancy between actual and expected rewards.

B.5 Task setups

GSM8K is a benchmark for mathematical reasoning that requires multi-step problem solving. Given a context description and a question, it requires step-by-step mathematical reasoning and computation to arrive at a final answer. The dataset contains approximately 7.5K problems in the training set and 1.3K problems in the test set. Problems range from basic arithmetic to complex word problems, testing both mathematical and logical reasoning capabilities.

SVAMP is a challenging mathematical word problem dataset specifically designed to test the robustness of language models in solving arithmetic problems. It contains 1,000 elementary math word problems, carefully curated to probe for specific vulnerabilities in mathematical reasoning systems. The problems require understanding both mathematical concepts and natural language semantics, with a focus on structural variations that test genuine problem-solving capabilities rather than pattern matching.

Strategy QA is a question answering dataset that focuses on multi-hop reasoning and strategic thinking. It consists of 2,290 yes/no questions, each requiring implicit multi-step reasoning and background knowledge to arrive at the correct answer. Unlike traditional QA datasets, Strategy QA questions cannot be answered by simply retrieving and combining explicit facts, making it an effective benchmark for testing complex reasoning capabilities.

MATH is a comprehensive mathematics dataset spanning various topics from algebra to calculus. It contains approximately 12K problems across different difficulty levels, with detailed step-by-step solutions. The dataset is structured into multiple categories including algebra, counting and probability, geometry, intermediate algebra, number theory, prealgebra, and precalculus, making it particularly effective for evaluating mathematical problem-solving capabilities across different domains.

GSM-Hard is a specialized subset of mathematical word problems specifically designed to test advanced reasoning capabilities. It contains problems that are significantly more challenging than standard GSM8K problems, requiring more complex multi-step reasoning and mathematical operations. The dataset focuses on problems that typically have lower success rates with standard approaches, making it particularly useful for evaluating the upper bounds of model performance.

TravelPlanner is a benchmark crafted for evaluating language agents in tool-use and complex planning within multiple constraints. The dataset comprises 1,225 queries in total, divided into training (45 queries), validation (180 queries), and test (1,000 queries) sets. The benchmark incorporates three types of constraints: environment constraints for testing adaptability to real-world conditions, commonsense constraints for evaluating practical reasoning, and hard constraints for assessing the ability to satisfy specific user requirements such as budget limitations. This structure makes TravelPlanner particularly effective for evaluating both reasoning capabilities and practical planning skills in real-world scenarios.

B.6 HYPERPARAMETER

Table 8: Hyperparameters of EcoNash

Parameter	Value	Description
Training Configuration		
Episodes per Task	100	Number of episodes per task
Buffer Size	32	Size of on-policy buffer
Batch Size	16	Mini-batch size for training
Update Interval	8	Policy update frequency (episodes)
Optimizer	Adam	Optimization algorithm
Learning Rate (η)	0.001	Learning rate for execution LLMs
Learning Rate (η_{coord})	0.0005	Learning rate for coordinator LLM
Discount Factor (γ)	0.99	Discount factor for future rewards
Network Architecture		
Entity Dimension (d)	256	Dimension of entity embeddings
Belief State Dimension (d_b)	128	Dimension of belief state
Attention Heads (H)	4	Number of attention heads
MLP Hidden Size	256	Hidden layer size in belief encoder
Transformer Blocks	2	Number of transformer layers
Key/Query Dimension	64	Dimension per attention head (d/H)
Feed-forward Size	1024	Dimension of FFN intermediate layer
Dropout Rate	0.1	Dropout probability in attention
Layer Norm Epsilon	1×10^{-5}	Layer normalization parameter
Temperature and Sampling Control		
T_{min}	0.1	Minimum temperature value
T_{max}	2.0	Maximum temperature value
p_{min}	0.1	Minimum sampling parameter
p_{max}	0.9	Maximum sampling parameter
Reward Configuration		
R_{max}	1.0	Maximum reward bound
α_1 (AL weight)	0.4	Action Likelihood reward weight
α_2 (TS weight)	0.4	Task-specific reward weight
α_3 (SE weight)	0.2	Self-Evaluation reward weight
Loss Weights		
λ_b	0.1	Weight for belief network loss
λ	0.1	Regularization weight in encoder
λ_m	0.1	Weight for mixing network consistency
Early Stopping		
ϵ_C	0.01	Commitment change threshold
ϵ_L	1×10^{-4}	Loss convergence threshold
$R_{\text{threshold}}$	0.7	Average reward threshold
T_{patience}	5	Patience epochs for validation
Model Size		
Learnable Parameters	$\sim$1.7M	Total trainable parameters

Coordinator Prompt(for Strategy)

"You are a coordinator in a multi-agent system responsible for devising effective strategies to solve a given problem. Based on the following problem, generate a concise high-level strategy in English, no more than 50 tokens:
Problem: {question}
Please provide a strategy considering the following points:
1.Key elements and objectives of the problem
2.Possible solutions or steps
3.Potential challenges or limitations
4.Key aspects to focus on
Strategy:"

Figure 4: Coordinator Prompt(for Strategy)

C Prompt

D Example

D.1 Case study

Coordinator Prompt(for Commitment)

"You are a coordinator in a multi-agent system responsible for reviewing the answers of multiple execution LLMs based on a given strategy. Your tasks are:
1.Form a Commitment: Integrate the best aspects of all answers to ensure consistency in the solution process and accuracy in the final result.
2.Evaluate each answer: Assess the similarity of the solution process to the Commitment and the accuracy of the final result. Based on these criteria, assign a reward score between 0 and 1 to each answer.
Strategy: {strategy}
Execution LLMs' Answers:
•LLM1: {answer1}
•LLM2: {answer2} ...

•LLMn: {answern}

Please follow these steps: a. Review each LLM's answer to determine its adherence to the strategy and the correctness of the solution. b. Formulate a comprehensive Commitment by integrating the most effective methods and accurate results from the answers. c. Evaluate each answer based on the following criteria:
•Process Similarity: The consistency of the solution steps with the Commitment
•Result Accuracy: The correctness of the final answer Assign a reward score between 0 and 1 to each LLM, where 1 means full adherence to the Commitment and completely correct results, and 0 means no adherence or incorrect results. Please output the results in the following structured format:

Commitment: {Detail the integrated solution here, including key steps and the final result}
Evaluation and Rewards:
•LLM1: {score1} (Brief explanation for the score no more than 10 tokens)
•LLM2: {score2} (Brief explanation for the score no more than 10 tokens)
•...
•LLMn: {scoren} (Brief explanation for the score no more than 10 tokens)

Figure 5: Coordinator Prompt(for Commitment)

Execution LLM

"You are an execution LLM in a multi-agent system, responsible for deriving solutions based on a given strategy and your own belief network. Each LLM has different beliefs but cannot access the outputs of other LLMs. Your tasks are:
1.Form your belief based on the strategy: Assume other LLMs will follow certain potential solutions. Your goal is to generate the optimal solution without global information.
2.Output the best answer: Considering your belief about other LLMs' outputs, derive the optimal solution for the current environment.
3.Bayesian Nash Equilibrium: Your output should maximize expected utility under incomplete information, aligning with the strategy.
4.Feedback adjustment: Ensure your solution is coherent under uncertainty and optimized for the best result.
Strategy: {strategy}
Please follow these steps: a. Review the strategy and form your belief on how other LLMs might output. b. Based on your belief, derive and output your optimal solution. c. Ensure your solution aligns with Bayesian Nash Equilibrium, maximizing expected utility.
Final answer:"

Figure 6: Execution LLM

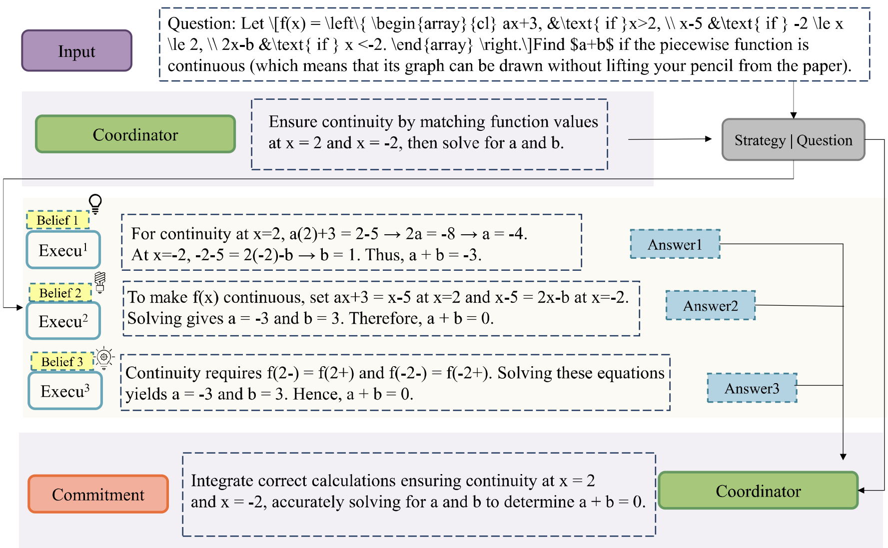

Figure 7: case study of math

D.2 Strategy Example

D.2.1 GSM8K

Q1: John buys 3 pizzas for \$12 each. If he gives the delivery person a 20% tip on the total, how much did he spend in total?

S1: Calculate pizza subtotal first. Add 20% of subtotal for tip. Sum for final amount.

F1:

1. Pizza cost = \$? $\times$?

2. Tip = ? $\times$ subtotal
3. Total = subtotal + tip

$Strategy + Format$: $35 tokens$

Q2: Janet saves twice as much money as Tom. If Tom saves $45 per week, how much does Janet save in 5 weeks?

S2: Find Janet's weekly savings relative to Tom's. Multiply by number of weeks.

F2:

1. Janet weekly = ? $\times$ Tom
2. Total = weekly $\times$ weeks

$Strategy + Format$: $28 tokens$

Q3: A factory produces 150 cars per day. If they increase production by 15% next month, how many cars will they produce in a 30-day month?

S3: Calculate production increase. Add to original. Multiply by days in month.

F3:

1. Increase = original $\times$ 15%
2. New daily = original + increase
3. Monthly = daily $\times$ days

$Strategy + Format$: $36 tokens$

Q4: Alex has 240 marbles and gives $\frac{3}{8}$ of them to Sarah. Sarah then gives $\frac{1}{4}$ of her marbles to Tom. How many marbles does Sarah have left?

S4: Calculate Sarah's initial share. Find amount she gives to Tom. Subtract.

F4:

1. Sarah gets = total $\times$ $\frac{3}{8}$
2. Sarah gives = her marbles $\times$ $\frac{1}{4}$
3. Remaining = initial - given

$Strategy + Format$: $39 tokens$

Q5: A train travels at 60 mph for 2.5 hours, then increases speed to 75 mph for 1.5 hours. What's the total distance traveled?

S5: Calculate distance for each speed separately using $d = r \times t$. Sum distances.

F5:

1. First distance = speed$_1$ $\times$ time$_1$
2. Second distance = speed$_2$ $\times$ time$_2$
3. Total = $d_1 + d_2$

$Strategy + Format$: $36 tokens$

D.2.2 MATH

Q1: In a bag of marbles, $\frac{3}{7}$ are blue and $\frac{2}{5}$ are red. The remaining 11 marbles are green. How many marbles are in the bag?

S1: Convert fractions to common denominator. Find the fraction for remaining color. Use given count to find total.

F1:

1. Convert to common denominator
2. Add converted fractions
3. Subtract from whole
4. Use remaining count to find total

$Strategy + Format : 32 tokens$

Q2: Find the area of a triangle with vertices at (0,0), (4,0), and (2,5).

S2: Use coordinate geometry method for area. Set up calculation matrix. Take final result.

F2:

1. Set up coordinate matrix
2. Calculate determinant
3. Apply area formula

$Strategy + Format : 28 tokens$

Q3: If $\log_2(x) = 3$ and $\log_2(y) = 4$, find $\log_2(xy)$.

S3: Apply logarithm properties. Combine given values. Express final result.

F3:

1. Write multiplication property
2. Substitute given values
3. Simplify result

$Strategy + Format : 26 tokens$

Q4: A circle has radius 6. Find the area of the sector formed by a 40° angle at the center.

S4: Convert angle measurement. Apply sector area formula. Simplify result.

F4:

1. Convert to radians
2. Write sector formula
3. Calculate final area

$Strategy + Format : 27 tokens$

Q5: Solve the equation: $2x^2 + 5x - 12 = 0$.

S5: Identify quadratic components. Apply standard formula. Solve for variables.

F5:

1. Identify coefficients
2. Setup quadratic formula
3. Calculate solutions

$Strategy + Format : 28 tokens$

D.2.3 SVAMP

Q1: There are 56 books on the shelf. Tom puts 14 more books and Jane removes 22 books. How many books are on the shelf now?

S1: Track sequential changes. Apply additions and subtractions in order.

F1:

1. Add new books
2. Subtract removed books

$Strategy + Format : 25 tokens$

Q2: A box has 3 rows of chocolates. Each row has 4 chocolates. If 5 chocolates were eaten, how many are left?

S2: Calculate initial total. Subtract consumed amount.

F2:

1. Find total chocolates
2. Subtract eaten ones

$Strategy + Format : 23 tokens$

Q3: Mary has 5 times as many stickers as John. John has 12 stickers. How many stickers do they have together?

S3: Calculate second person's amount. Sum both quantities.

F3:

1. Find Mary's stickers
2. Add both totals

$Strategy + Format : 24 tokens$

Q4: A garden has 35 flowers. ($\frac{2}{7}$) are roses and ($\frac{3}{7}$) are tulips. How many flowers are neither roses nor tulips?

S4: Sum known fractions. Find remaining fraction. Calculate final count.

F4:

1. Add type fractions
2. Find remaining fraction
3. Calculate flower count

$Strategy + Format : 27 tokens$

Q5: Each child needs 3 pencils. If there are 23 children, how many boxes of 10 pencils should the teacher buy?

S5: Calculate total need. Convert to required units. Round appropriately.

F5:

1. Calculate total pencils
2. Divide by box size
3. Round to whole boxes

$Strategy + Format : 28 tokens$

Note on Token Counts:

- All problems now follow consistent format: strategy + step-by-step format
- Strategy statements aim to be concise yet clear
- Format points provide framework without giving solutions
- Token ranges:
  - Shortest: 23 tokens (SVAMP Q2)
  - Longest: 39 tokens (GSM8K Q4)
  - Average: ($\sim$)30 tokens