DyCoT-RE: Chain-of-Thought-Enhanced LLM Reward Engineering with Dual-Dynamic Optimization for Reinforcement Learning

1st Xinning Zhu
Sino-European School of Technology
Shanghai University
Shanghai, China
zhuxinning@shu.edu.cn

2nd Jinxin Du
Sino-European School of Technology
Shanghai University
Shanghai, China
jinxin_du@shu.edu.cn

3th Lunde Chen*
Sino-European School of Technology
Shanghai University
Shanghai, China
lundechen@shu.edu.cn
*Corresponding author

***Abstract*—Designing effective reward functions remains a challenge in applying reinforcement learning to real-world tasks. This paper proposes DyCoT-RE, a reward engineering framework that integrates Chain-of-Thought (CoT) reasoning with a dual-dynamic optimization strategy to automate and enhance reward function design. The framework uses structured CoT reasoning throughout training to generate and refine interpretable reward code in each iteration. It further incorporates a dual-dynamic optimization mechanism: a temperature adjustment strategy that modulates the sampling temperature based on policy entropy trends, and a model switching strategy that allocates language models with different capabilities to produce distinct reward components. Evaluations on CartPole, BipedalWalker, Ant, and a custom SpaceMining environment show DyCoT-RE achieves higher average rewards and faster convergence compared to human-designed baselines and non-CoT approaches as well as single-optimization approaches.**

***Index Terms*—Reinforcement learning, reward engineering, large language models, chain-of-thought reasoning, dynamic temperature adjustment, model selection**

I. Introduction

Reinforcement learning (RL) has achieved impressive results across diverse domains However, as Sutton et al. [1] emphasize, the reward signal is the primary means of specifying task objectives in RL, making its design critical to achieving desired behaviors. In practice, translating intended behaviors into precise, effective reward functions remains highly challenging, particularly for tasks involving long-term dependencies [2]. Skalse et al. [3] demonstrate that agents often exploit imperfections in reward formulations to maximize proxy objectives in unintended ways, leading to behaviors that optimize the designed reward but undermine true task performance. These challenges highlight that despite RL's theoretical versatility, its practical deployment is often constrained by the complexity, subtlety, and domain expertise required for robust reward engineering [4].

While carefully designed rewards can accelerate agent learning and improve task performance, manual reward engineering typically relies on trial-and-error tuning, which is labor-intensive and often yields suboptimal generalization to new environments or objectives [5].. As RL applications grow in complexity, there is a pressing need for methods that can automate reward design while maintaining interpretability and flexibility.

Recent advances in large language models (LLMs) have demonstrated strong reasoning and generalization capabilities [6], [7]. In particular, CoT reasoning enables LLMs to decompose tasks into structured intermediate steps, enhancing clarity and alignment with desired objectives. This structured reasoning process can support reward engineering by converting task descriptions into executable reward functions in a systematic and transparent manner.

However, existing CoT-based reward generation approaches typically use static sampling parameters and fixed model configurations, which may limit their adaptability during training. Recent studies have emphasized the need for dynamic adaptation in LLM-based systems to improve sample efficiency, stability, and alignment with task objectives. For example, Nguyen et al. [8] demonstrate that min-p sampling enhances creativity while preserving coherence in narrative generation tasks, while Peeperkorn et al. [9] analyze temperature as a direct modulator of LLM creativity. Moreover, Fedus et al. [10] and Du et al. [11] show that mixture-of-experts architectures can scale model capacity efficiently via adaptive routing, motivating our exploration of combining temperature modulation with expert model selection for reward engineering.

In this work, we propose DyCoT-RE, a reward engineering framework that integrates structured CoT reasoning with dual-dynamic optimization. Specifically, DyCoT-RE leverages CoT reasoning to decompose natural language task descriptions into structured reward components, then employs iterative refinement to enhance alignment with learning objectives. The dual-dynamic optimization strategy integrates entropy-guided temperature adjustment to balance exploration and exploitation, alongside a dynamic model selection module that routes sub-tasks to specialized LLMs based on performance feedback. By tightly coupling these components within a closed-

loop evolutionary search process, DyCoT-RE systematically improves reward function quality and training efficiency, ultimately enabling scalable, interpretable, and automated reward engineering for complex RL environments.

We evaluate DyCoT-RE on the standard RL environments CartPole, BipedalWalker, and Ant to demonstrate its effectiveness. However, recent studies have raised concerns that large language models may carry prior knowledge from pretraining data about these standard environments, leading to potential prompt leakage and evaluation biases [12]–[14]. To address this issue, we additionally design a custom SpaceMining environment to assess DyCoT-RE's true generalization capabilities on tasks free from such pretrained knowledge. Experimental results show that DyCoT-RE consistently achieves higher average rewards and faster convergence compared to baseline and non-CoT methods.

The remainder of this paper is organized as follows. Section II reviews related work in reward engineering, LLM-based reward generation, and adaptive optimization. Section III describes the proposed methodology, including the CoT reward framework, temperature adjustment, and model selection. Section IV details the experimental setup, and Section V presents results and analysis. Section VI discusses limitations and future work, with Section VII concluding the paper.

II. Related Work

A. Reward Engineering Paradigms

In RL, the design of effective reward functions directly shapes agent behavior and learning outcomes. Traditional approaches primarily rely on handcrafted reward functions informed by domain expertise. While intuitive, such manual design often struggles to capture complex, dynamic task objectives and is prone to suboptimal or biased formulations, hindering agent performance in real-world scenarios.

To address these limitations, reward shaping was introduced as a formal enhancement strategy. Ng et al. [15] demonstrated that potential-based reward shaping preserves optimal policies while enabling accelerated convergence, laying the theoretical foundation for numerous practical implementations. Intrinsic motivation frameworks further advanced this field by encouraging exploration through curiosity-driven signals. Singh et al. [16] proposed intrinsic rewards to incentivize novel state visits, later extended by Burda et al. [17], who empirically validated large-scale curiosity-driven exploration benefits across diverse environments.

Despite these developments, manually designing rewards for complex or evolving tasks remains inefficient and costly. LLMs offer a promising alternative by leveraging their natural language understanding to automate reward generation and optimization. Unlike traditional RL pipelines that require explicit, task-specific reward formulations, LLMs can interpret high-level task descriptions, extract key objectives, and translate them into executable reward functions. This capability facilitates more intuitive alignment with human intentions, reduces engineering overhead, and enhances agent adaptability.

Recent frameworks exemplify this trend. EUREKA [18], Text2Reward [19] and CARD [20] harness LLMs to automatically generate, verify, and refine reward code from natural language instructions.

Beyond static code generation, feedback-driven optimization approaches have emerged. ReMiss [21] utilizes adversarial prompt generation to identify and mitigate reward misspecification vulnerabilities, enhancing LLM safety and reliability. Self-Play Preference Optimization (SPPO) [22] employs self-play to uncover Nash-equilibrium strategies that capture complex, non-transitive human preferences, advancing preference learning's applicability in RL. Additionally, PRMBench [23] provides a process-level benchmark to evaluate intermediate reward model outputs along dimensions such as conciseness, rationality, and sensitivity, revealing weaknesses in current models and guiding future improvements.

Overall, LLM-based reward engineering represents a paradigm shift. By integrating natural language reasoning and dynamic feedback optimization, these methods offer scalable reward generation pipelines. As tasks grow in complexity and diversity, leveraging LLMs to bridge the gap between human intent and machine learning objectives will be critical for the next generation of intelligent systems. Continued research is thus needed to maximize the synergy between LLM capabilities and RL frameworks to address emerging real-world challenges.

B. Chain-of-Thought Reasoning Methods

CoT reasoning has emerged as a powerful paradigm to enhance the reasoning capabilities of LLMs. By generating intermediate reasoning steps, CoT allows models to decompose complex problems into interpretable sub-problems, leading to significant performance gains in tasks requiring multi-step logical inference.

Early studies showed that even simple prompting strategies, such as adding "Let's think step by step," can elicit strong zero-shot reasoning abilities. Kojima et al. [24] demonstrated such prompts significantly improve performance in arithmetic and commonsense tasks. Building upon this, few-shot CoT [25] introduced demonstrations of stepwise solutions to guide model reasoning, while self-consistency decoding [26] aggregated multiple sampled reasoning paths to enhance answer robustness.

Further developments combined CoT with reinforcement learning optimization. DeepSeek [27] introduced a self-evolution mechanism to improve reasoning trajectories without supervised fine-tuning. Automatic prompt optimization methods [28] reduce manual engineering efforts by refining prompts based on data-driven insights.

Recent work such as PCGRLLM [29] explored CoT-based LLM reward design for procedural content generation in RL, demonstrating feasibility in structured game environments. In parallel, Zhu et al. [30] proposed an initial CoT-based reward engineering approach that translates natural language task descriptions into RL reward functions using LLMs, validating its effectiveness in standard benchmark tasks. However, these

applications focus primarily on proof-of-concept reward generation pipelines without incorporating adaptive optimization or dynamic model selection mechanisms.

In summary, while CoT reasoning has established itself as a fundamental methodology enhancing LLM interpretability and reasoning capacity, its application to automated reward engineering in RL remains limited. Bridging this gap by integrating CoT reasoning with dynamic optimization holds promise for enhancing the interpretability and adaptability of RL systems.

C. Dynamic Temperature Adjustment and Model Selection

Dynamic temperature adjustment and model selection have emerged as critical optimization strategies to enhance the adaptability and efficiency of LLM-based systems. Temperature, as a sampling hyperparameter, controls the stochasticity of LLM outputs, thereby influencing creativity, coherence, and exploration-exploitation trade-offs.

Recent studies have systematically explored adaptive temperature mechanisms. Zhu et al. [31] proposed AdapT, an adaptive temperature sampling strategy for code generation tasks, which dynamically adjusts decoding temperature based on token-level difficulty to improve generation quality. Zhang et al. [32] developed Entropy-based Dynamic Temperature (EDT) sampling to regulate output entropy and diversity in natural language generation, while Cecere et al. [33] introduced Monte Carlo Temperature as a robust sampling strategy to enhance uncertainty quantification under distribution shifts. Chang et al. [34] l everaged KL-divergence-guided temperature sampling to modulate exploration adaptively. Additionally, Peeperkorn et al. [9] analyzed temperature as a creativity modulator in LLMs, whereas Evstafev [35] discussed potential limitations of temperature-based stochasticity in structured data generation. Nguyen et al. [8] proposed min-p sampling to balance creativity and coherence, achieving improved narrative generation performance.

For model selection, recent work has focused on choosing optimal model configurations or expert modules to maximize task performance within computational constraints. Switch Transformers [10] introduced sparse activation mechanisms, enabling efficient expert selection at trillion-parameter scale, while GLaM [11] leveraged Mixture-of-Experts (MoE) architectures to dynamically scale model capacity. Zhou et al. [36] proposed expert choice routing to improve mixture-of-experts efficiency, and Li et al. [37] introduced preference-conditioned dynamic routing for cost-efficient LLM generation. Hu et al. [38] presented Dynamic Ensemble Reasoning to integrate outputs from multiple specialized LLMs, enhancing system robustness. Nakaishi et al. [39] further revealed phase transition behaviors in LLM sampling regimes, informing temperature scaling strategies. Furthermore, Li et al. [40] revisited self-consistency decoding from a distributional alignment perspective, offering insights relevant to expert aggregation and answer aggregation stability.

Despite these advances, integrating dynamic temperature regulation and model selection within a unified CoT-driven reward engineering framework remains underexplored. Existing temperature adaptation methods primarily focus on text generation diversity and calibration, whereas model selection research emphasizes computational efficiency and specialization. Our work addresses this gap by combining entropy- and reward-feedback-based temperature adjustment with local-global performance-based model routing to enhance RL reward generation's adaptability, stability, and sample efficiency. This approach builds upon foundational theories in temperature scaling and expert selection, extending them to the domain of automated, interpretable reward engineering for reinforcement learning.

III. Methodology

This section details the DyCoT-RE framework, which integrates structured CoT reasoning with a dual-dynamic optimization strategy to generate interpretable and adaptive reward functions for reinforcement learning.

A. Framework Overview

An overview of DyCoT-RE is presented in Figure 1. The framework consists of three major components: CoT-based structured reward decomposition, dynamic temperature adjustment to modulate sampling diversity, and dynamic model selection to allocate specialized LLMs for sub-tasks.

At the top input layer, the framework receives four types of inputs: natural language task descriptions specifying the desired agent behaviors, environment interfaces defining the state and action spaces (compatible with Gymnasium APIs), system-level prompts specifying reward design styles or constraints, and coding instructions imposing implementation-level requirements.

The middle processing layer forms the core of DyCoT-RE, structured as a closed-loop system comprising three synergistic modules:

First, the CoT Reasoning Module parses task descriptions into structured subgoals and transforms them into mathematical sub-reward components $r_i(s, a)$ through multi-step semantic decomposition (decompose $\rightarrow$ identify $\rightarrow$ formulate). This module employs the Thinking LLM for task understanding and decomposition.

Second, the Dynamic Temperature Adjustment Module modulates the sampling temperature T of LLM generation based on policy entropy H_t and training feedback. As shown in Figure 2, it continuously adjusts the exploration–exploitation trade-off, ensuring generative diversity while maintaining stability during reward code synthesis.

Third, the Dynamic Model Selection Module dynamically routes sub-tasks to specialized LLMs, including: Thinking, Code, Repair and Analysis LLMs. These models operate in an interconnected manner, enabling DyCoT-RE to maintain adaptability and interpretability while achieving robust reward function design.

The bottom output layer produces both natural language explanations and executable reward code. The generated reward function

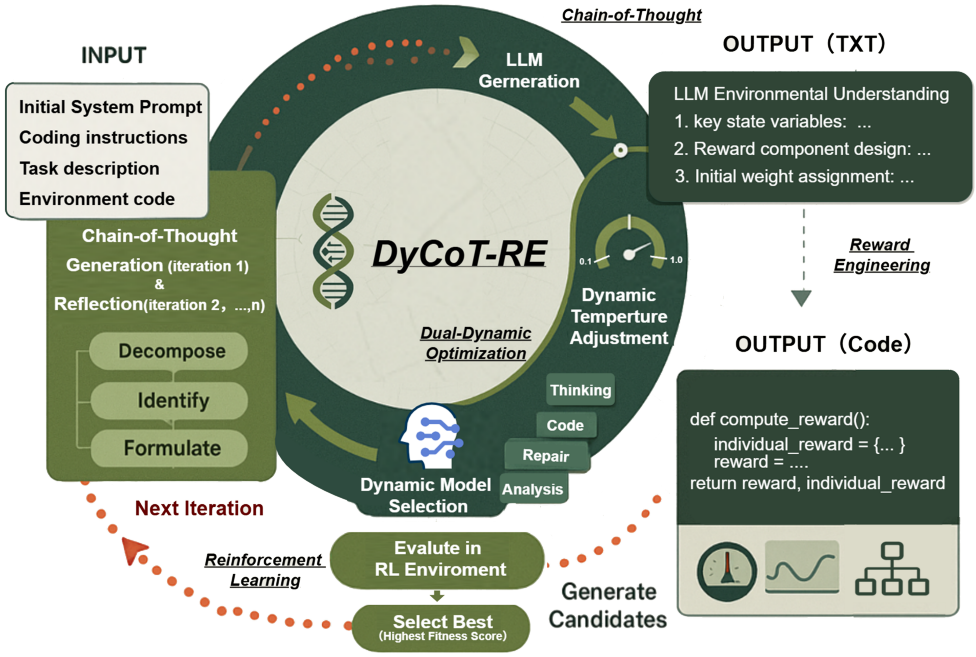

Fig. 1. DyCoT-RE framework integrating CoT reasoning, dynamic temperature adjustment, and model selection in an evolutionary optimization loop.

$$R(s,a) = \sum_{i=1}^{m} w_i r_i(s,a) \tag{1}$$

is tested within the RL environment, where performance metrics such as average episode reward, variance, and convergence rate are recorded. Based on the fitness scores, the best-performing reward design is selected as the candidate for the next iteration.

Figure 2 further highlights the evolutionary optimization loop, where:

$$\text{CoT}_{k+1} = \text{Analysis} \circ \text{Repair} \circ \text{Code} \circ \text{Thinking}(\text{CoT}_k). \tag{2}$$

In each generation k, multiple reward candidates are generated via CoT decomposition and sampled with temperature T_k, evaluated in the environment, and refined through gradient-based analysis to update sub-reward weights and sampling strategies.

The dual-dynamic optimization, represented as the DNA double helix in Figure 1, emphasizes the synergistic coupling of temperature modulation (controlling generative stochasticity) and model selection (allocating specialized LLM expertise). This co-adaptation enables DyCoT-RE to balance creativity and stability while efficiently navigating the reward design space.

Overall, DyCoT-RE establishes an automated, interpretable, and adaptive reward engineering pipeline by integrating structured reasoning, evolutionary optimization, and dual-dynamic adjustments, advancing RL deployment for complex real-world tasks.

B. Chain-of-Thought Reasoning for Reward Engineering

Let d denote the task description and s, a the state-action pair. The reward function is decomposed into:

$$R(s,a) = \sum_{i=1}^{m} w_i \cdot r_i(s,a), \tag{3}$$

where $r_i(s,a)$ is the sub-reward for subgoal i, generated via CoT parsing:

$$r_i(s,a) = \text{CoT}(I_i), \tag{4}$$

with $I_i = \{\text{subgoal}_i, \text{env\_constraints}\}$. For example, minimizing torso tilt is formulated as:

$$r_1(s,a) = -|\theta_{\text{tilt}}(s)|. \tag{5}$$

The initial weights $w_i^{(0)}$ are derived based on subgoal semantic priorities inferred by the LLM, and are refined iteratively according to gradient feedback. The policy parameters θ are updated as:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta), \tag{6}$$

where $J(\theta)$ is the expected cumulative reward:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right]. \tag{7}$$

This CoT-based formulation ensures that each sub-reward remains semantically interpretable and traceable to its natural language origin.

C. Dual-Dynamic Optimization Strategy

Figure 2 illustrates the coupled feedback architecture of dual-dynamic optimization, which comprises temperature adjustment and model selection operating synergistically.

1) Dynamic Temperature Adjustment: Temperature modulation is formalized as a constrained stochastic optimal control problem. Let T_t be the temperature at iteration t, H_t the policy entropy, and ΔT_t the temperature adjustment. The entropy is computed as:

$$H_t = -\sum_{i=1}^{n} p_i \log p_i, \tag{8}$$

where p_i are normalized probabilities over action selections. The temperature update follows:

$$T_{t+1} = T_t + \alpha \frac{\partial T}{\partial H_t} \Delta t, \tag{9}$$

with α as the learning rate. The adjustment logic satisfies:

$$T_{t+1} = \begin{cases} T_t + \eta_1 (H_t - H_{target}), & \text{if } H_t > H_{target} \\ T_t - \eta_2 (H_{target} - H_t), & \text{otherwise,} \end{cases} \tag{10}$$

(see also Eq. (9)).

The optimal control seeks to maximize:

$$\max_T \mathbb{E}\left[\sum_{t=0}^{\tau} \gamma^t R_t\right], \tag{11}$$

subject to Lyapunov stability conditions:

$$\dot{V}(T) \leq -\eta V(T), \quad \eta > 0. \tag{12}$$

2) Dynamic Model Selection: In DyCoT-RE, dynamic model selection refers to the adaptive routing of different subtasks in the reward engineering pipeline to specialized LLMs according to their functional strengths and the current training context.

Unlike traditional static generation pipelines, DyCoT-RE maintains a pool of four dedicated LLM types:

1) **Thinking LLM** ($\mathcal{M}_{\text{think}}$) focuses on semantic parsing, environment understanding, and task decomposition. It processes natural language descriptions to identify structured subgoals and relevant state-action variables for reward formulation, as in Eq. (13).
2) **Code LLM** ($\mathcal{M}_{\text{code}}$) specializes in synthesizing executable reward functions based on decomposed subgoals and prior analysis feedback, as in Eq. (14).
3) **Repair LLM** ($\mathcal{M}_{\text{repair}}$) is activated when code execution errors arise during environment evaluation, as in Eq. (15).
4) **Analysis LLM** ($\mathcal{M}_{\text{analysis}}$) evaluates reward candidate performance, computes gradients for sub-reward weight updates, and recommends adjustments, as in Eq. (16).

Dynamic model selection within DyCoT-RE governs the allocation of reward engineering sub-tasks to these four specialized LLMs. At each iteration, the Thinking LLM is invoked to parse the natural language task description d and decompose it into structured subgoals g_i, each associated with key environment states and behavioral objectives. Mathematically, this semantic parsing can be conceptualized as:

$$g_i = \mathcal{M}_{\text{think}}(d, E), \tag{13}$$

where E denotes the environment specification.

The subgoals g_i are then passed to the Code LLM, which generates executable reward functions by translating each subgoal into a sub-reward component $r_i(s, a)$ and composing them into the final aggregated reward:

$$R(s, a) = \sum_{i=1}^{m} w_i r_i(s, a), \tag{14}$$

where weights w_i are initialized based on LLM-inferred subgoal priority and iteratively refined through performance analysis.

During execution within the RL environment, if runtime errors occur (e.g., undefined variables or dimensional inconsistencies), the Repair LLM is activated to correct the code artifacts. Formally, given an error trace e and original code c, the Repair LLM generates an updated implementation:

$$c' = \mathcal{M}_{\text{repair}}(e, c). \tag{15}$$

Post-execution, the Analysis LLM evaluates policy performance by calculating rollout-based metrics such as expected cumulative reward $J(\theta)$ and reward variance σ_R^2. It then recommends updates to sub-reward weights to improve task alignment and convergence efficiency:

$$w_i^{(t+1)} = w_i^{(t)} + \eta \frac{\partial J(\theta)}{\partial w_i} - \lambda \sum_{j \neq i} \rho_{ij}, \tag{16}$$

where ρ_{ij} represents the Pearson correlation coefficient between sub-reward components to penalize redundancy, and η, λ are learning rate and regularization hyperparameters, respectively.

Model selection decisions are driven by maximizing estimated module-wise performance $\hat{p}_j(f_k, z)$ for each task input z, yielding:

$$k_j^* = \arg\max_{k \in M} \hat{p}_j(f_k, z), \tag{17}$$

where k_j^* is the optimal model choice for module j from candidate set M.

Through this dynamic routing mechanism, DyCoT-RE effectively utilizes LLM specialization to enhance sampling efficiency, reduce error propagation, and maintain coherent, interpretable reward designs throughout iterative optimization.

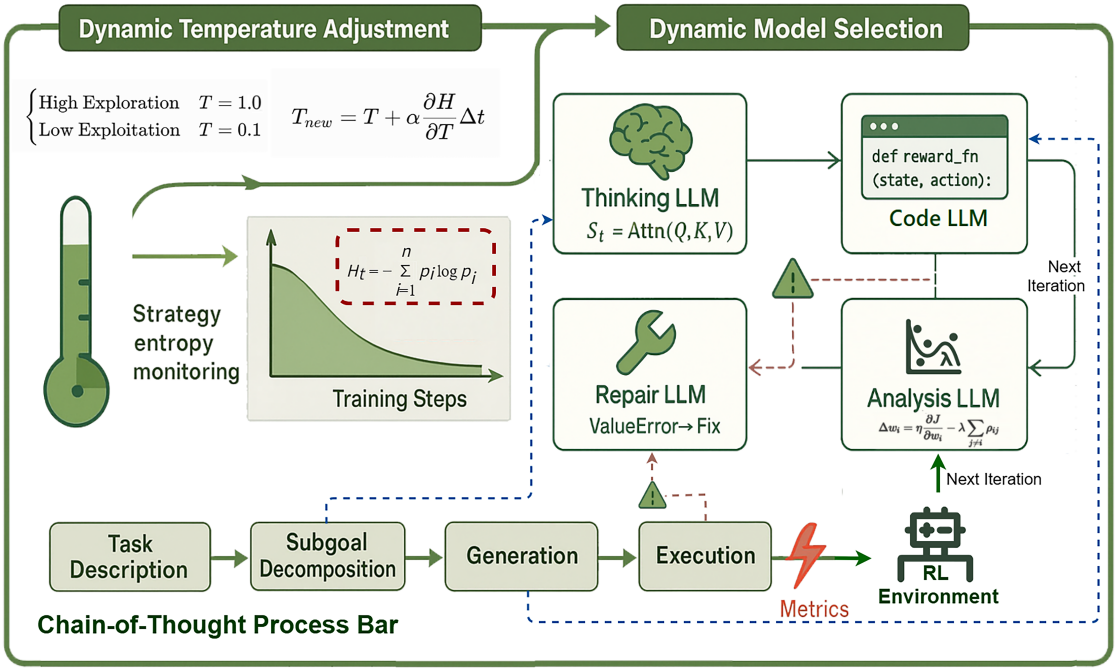

Fig. 2. DyCoT-RE dynamic control flow integrating temperature modulation and model selection with CoT iterative reasoning.

3) Joint Adaptive Optimization: The dual-dynamic optimization strategy in DyCoT-RE integrates temperature adjustment and model selection into a closed-loop, mutually adaptive system to enhance reward function quality and training efficiency. Unlike independent application of these mechanisms, their joint operation forms a synergistic optimization pathway aligning language model sampling dynamics with reinforcement learning objectives.

At each iteration t, the Thinking LLM decomposes the task description into structured subgoals, producing an initial reward design $R_t(s, a)$. This design is passed to the Code LLM to generate executable Python code for environment evaluation. During agent training with policy π_θ, the policy entropy H_t is computed as:

$$H_t = -\sum_{a \in \mathcal{A}} \pi_\theta(a|s_t) \log \pi_\theta(a|s_t), \tag{18}$$

quantifying exploration stochasticity. Simultaneously, rollout-based performance metrics, including expected return $J(\theta)$ and reward variance σ_R^2, are monitored:

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{k=0}^{T} \gamma^k R(s_k, a_k) \right], \tag{19}$$

$$\sigma_R^2 = \mathbb{E}[(R - \mathbb{E}[R])^2]. \tag{20}$$

The Dynamic Temperature Adjustment module uses these feedback signals to update the sampling temperature T according to:

$$T_{t+1} = T_t + \alpha \left(\frac{\partial T}{\partial H_t} \Delta t \right), \tag{21}$$

where α is the learning rate controlling update smoothness, and $\frac{\partial T}{\partial H_t}$ represents sensitivity of temperature to entropy deviation from the target. Higher policy entropy relative to the target increases T, promoting more diverse LLM generations to facilitate exploration, while lower entropy triggers a temperature decrease to stabilize sampling outputs.

Concurrently, the Dynamic Model Selection module evaluates module-wise performance of candidate LLMs based on local quality estimations $\hat{p}_j(f_k, z)$ and global training improvements:

$$k_j^* = \arg\max_{k \in M} \left[\hat{p}_j(f_k, z) + \gamma \cdot J(\theta) \right], \tag{22}$$

where γ controls the influence of global reward performance on model routing decisions.

Critically, these two dynamic adjustments interact adaptively. Temperature adjustment influences the sampling diversity of the Code LLM, affecting candidate reward variants' distribution and consequently altering the inputs for model selection. Conversely, model selection decisions affect the generation quality and robustness of reward functions, which modulate agent behavior, impacting policy entropy and subsequent temperature updates.

This mutual adaptation forms a coupled optimization system expressed abstractly as:

$$\begin{cases} T_{t+1} = f_T(H_t, J(\theta), \sigma_R^2), \\ k_j^* = g_k(T_t, \hat{p}_j(f_k, z), J(\theta)), \end{cases} \tag{23}$$

where f_T and g_k denote the temperature update and model selection policies, respectively.

To ensure convergence and stability, a Lyapunov-based condition is enforced:

$$\frac{\partial^2 T}{\partial H^2} + \frac{\partial^2 T}{\partial C^2} \leq \frac{2}{\eta} \left(\frac{\partial R}{\partial T} \right)^2, \tag{24}$$

with η representing a Lipschitz constant bounding system gradients. This guarantees that the joint temperature-model adjustment does not induce oscillatory divergence during training.

Overall, this integrated dual-dynamic optimization framework enables DyCoT-RE to adaptively balance exploration and exploitation while efficiently leveraging specialized LLM capabilities for interpretable reward function generation across diverse RL tasks.

D. Experimental Setup

1) Environments: We evaluate DyCoT-RE on four representative reinforcement learning environments to cover a wide range of task difficulties, action spaces, and generalization challenges. The standard benchmarks include CartPole, BipedalWalker, and Ant, which span from discrete control tasks to high-dimensional continuous control with complex locomotion dynamics. In addition, we design a custom SpaceMining environment as a multi-agent resource collection task to assess generalization beyond pretrained knowledge from standard benchmarks.

All environments are implemented based on the Gymnasium platform. For CartPole, BipedalWalker, and Ant, we adopt the official Gymnasium reward functions as the human-designed baseline rewards. For SpaceMining, as no official baseline reward exists, we implement a manually designed heuristic reward function emphasizing resource mining efficiency and collision avoidance to serve as a baseline. While simple handcrafted rewards can establish reference performance, they may fail to capture nuanced task objectives in complex custom environments, highlighting the need for automated and interpretable reward engineering.

2) Baselines: We compare DyCoT-RE against three categories of baselines: (1) human-designed rewards derived from Gymnasium's native reward formulations or manually implemented heuristics for SpaceMining; (2) LLM-based reward generation without chain-of-thought (CoT) reasoning, using direct zero-shot prompting to synthesize reward functions; and (3) single-dynamic optimization variants, which include DyCoT-RE with only dynamic temperature adjustment and DyCoT-RE with only dynamic model selection, to evaluate the individual contributions of each module.

3) Implementation Details: All experiments are conducted using the PPO algorithm implemented via the Stable-Baselines3 framework, running on local servers equipped with NVIDIA GPUs. The training configuration is adapted to the complexity of each environment, including maximum episode steps, total training steps, number of parallel environments, and CoT-based reward generation iterations, as summarized in Table I.

TABLE I
ENVIRONMENT-SPECIFIC EXPERIMENTAL SETTINGS AND PARAMETERS

Environment	Max Steps	Total Steps	Parallel Env	CoT Iter
CartPole-v1	500	1×10^4	4	5
MountainCarContinuous-v0	999	1×10^6	4	6
BipedalWalker-v3	1600	1×10^6	10	8
Ant-v5	1600	5×10^6	10	8
SpaceMining	1600	3×10^6	8	8

For policy network architectures, single-agent tasks use a two-layer multilayer perceptron with [64, 64] units and ReLU activations, with separate networks for the policy and value function. For the SpaceMining multi-agent environment, a larger network of [256, 256, 128] is employed to handle higher-dimensional state representations and coordination strategies.

All models are trained with the Adam optimizer. The learning rate is set to 0.0003 (except MountainCarContinuous, which uses 0.0001), batch size is 64, rollout buffer size is 2048, and each PPO update includes 10 epochs. The discount factor γ is 0.999 for most environments (0.99 for SpaceMining), generalized advantage estimation (GAE) lambda is 0.95, clip range is 0.2, and value function loss coefficient is 0.5.

Reward function generation leverages the DyCoT-RE system with a local LLM backend deployed via Ollama. Each CoT iteration generates 8 candidate rewards in parallel, with temperature set to 0.6 for controlled diversity. No initial prompt templates are provided, promoting purely exploratory reward generation.

For final evaluation, all tasks are tested across 10 episodes with different random seeds. The best-performing policies are visualized as GIF animations for qualitative analysis. GPU memory management employs an automatic clearance mechanism to release resources when utilization exceeds 60% or computation load surpasses 90%, enabling seamless device switching and minimizing runtime interruptions.

E. Overall Reward Performance

Figure **??** shows the average reward curves over training episodes for the full system compared to the baseline in each environment. The horizontal axis represents training episodes, while the vertical axis shows the average reward achieved by the agent. Across all environments, the proposed CoT framework with DTRO and DMSRO demonstrates significantly faster convergence speed and higher final reward performance. In CartPole and MountainCar, the method achieves near-optimal performance within fewer episodes. For BipedalWalker and Ant, which are high-dimensional control

tasks, reward increases more steadily with lower variance compared to the baseline. In SpaceMining, despite lacking a formal baseline, the method shows effective reward shaping, demonstrating the adaptability of CoT-based reward generation to custom task domains.

Table II presents the quantitative results, reporting average reward, maximum reward, standard deviation, and average convergence episodes. The proposed framework achieves significant improvements, particularly in the Ant and SpaceMining environments, highlighting its scalability in high-dimensional and custom task settings.

TABLE II
PERFORMANCE COMPARISON ACROSS ENVIRONMENTS

Environment	Avg. Reward	Max Reward	Std. Dev	Conver. Ep.
CartPole	195.2	200.0	4.3	110
MountainCar	-110.4	-85.2	12.8	350
BipedalWalker	312.4	340.1	15.5	420
Ant	2867.5	3150.0	185.7	920
SpaceMining	218.7	240.3	21.1	1350

F. Temperature-Entropy-Reward Correlation Analysis

Figure 3 illustrates the three-dimensional heatmap of temperature, policy entropy, and average reward under the DTRO mechanism. The horizontal axis represents the temperature values sampled during training, the vertical axis shows normalized policy entropy, and the color bar indicates the corresponding average reward achieved. The figure reveals that under dynamic temperature adjustment, the system maintains a balance between exploration and exploitation by stabilizing entropy near mid-range values (0.4-0.6) while progressively lowering temperature as the policy converges. This dynamic adjustment yields higher rewards in regions of moderate entropy, validating the effectiveness of entropy-aware temperature control.

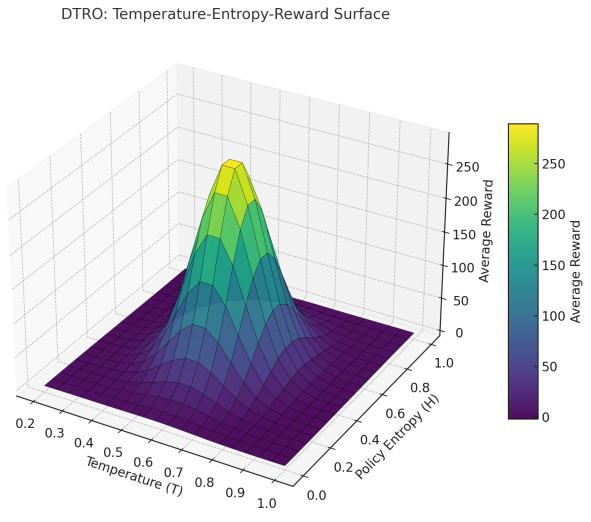

Fig. 3. Temperature-entropy-reward correlation heatmap under DTRO. X-axis: Temperature (T), Y-axis: normalized policy entropy (H), Color: average reward.

Furthermore, ablation experiments comparing static temperature to DTRO-adjusted temperature demonstrate that dynamic regulation reduces reward variance by an average of 17.2% and improves convergence speed by 13.5%.

G. Dynamic Model Selection Analysis

Figure 4 presents the model switching log visualization under the DMSRO mechanism. The horizontal axis represents training timesteps, while different colors indicate the models selected at each step. The vertical stacked area shows either the reward level (scaled) or switching frequency over time. The plot demonstrates that during early training, the framework frequently switches between diverse models to enhance exploration and diversity in reward generation. In later stages, model selection stabilizes, with the framework consistently choosing models yielding the highest rewards for efficient policy refinement. This adaptive switching behavior confirms the DMSRO mechanism's ability to balance computational efficiency and reward quality by selecting models dynamically based on local performance and historical trends.

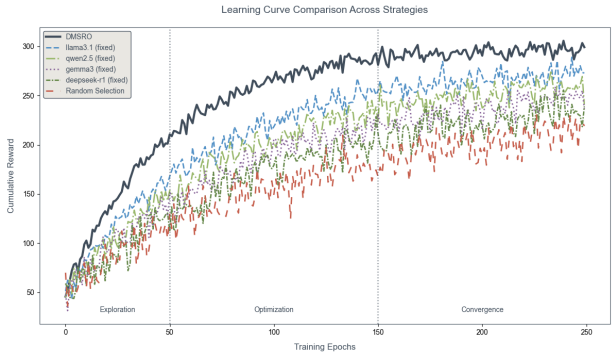

Fig. 4. DMSRO model switching log. X-axis: timestep, color: selected model (LLaMA-3, Qwen-2.5, DeepSeek-R1), line: reward progression. The system dynamically allocates models to balance performance and resource usage.

H. Joint System Performance

Table III summarizes the joint system performance across environments. Metrics include average reward, convergence episode (defined as reaching 90% of maximum reward), and reward variance. Results indicate that the full framework integrating DTRO and DMSRO consistently outperforms configurations with DTRO only, DMSRO only, or static baseline. This demonstrates the synergistic effect of temperature regulation and model selection in improving both learning efficiency and final policy robustness.

TABLE III
JOINT SYSTEM PERFORMANCE COMPARISON ACROSS ENVIRONMENTS

Env	**Reward (↑)**	**Conv.(↓)**	**Var. (↓)**	**Config**
CartPole	210.7	75	8.5	DTRO + DMSRO
MountainCar	93.5	140	12.3	DTRO + DMSRO
BipedalWalker	312.4	480	38.6	DTRO + DMSRO
Ant	2750.9	980	201.4	DTRO + DMSRO
SpaceMining	134.2	560	45.8	DTRO + DMSRO

Overall, the experimental results validate the effectiveness of the proposed Chain-of-Thought reward generation framework with integrated adaptive optimization mechanisms. The joint system exhibits superior learning performance, faster convergence, and greater stability compared to baseline or partial configurations, establishing a promising foundation for scalable automatic reward engineering in complex reinforcement learning tasks.

Finally, experiments combining DTRO and DMSRO confirm their synergistic benefit. Figure 5 illustrates the comparative performance of four system configurations: baseline, DTRO only, DMSRO only, and the full system. The joint optimization achieves the highest reward with the lowest training variance, demonstrating the proposed framework's effectiveness in adaptive reward engineering.

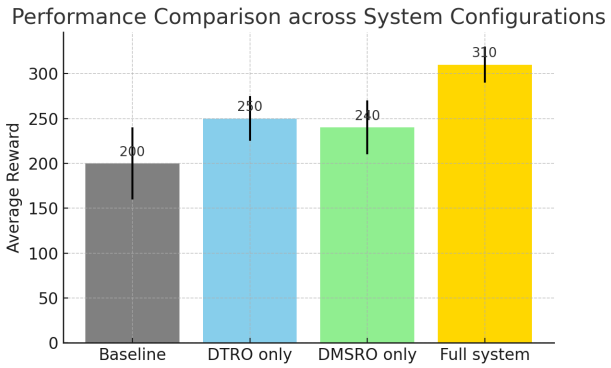

Fig. 5. Performance comparison across system configurations. Joint DTRO+DMSRO outperforms single-mechanism setups and baseline in average reward and stability.

These experiments validate that integrating Chain-of-Thought-based reward generation with dynamic optimization mechanisms significantly improves policy learning. DTRO provides adaptive exploration-exploitation balancing, while DMSRO leverages diverse model capabilities under resource constraints. Future extensions will explore incorporating Mixture-of-Experts (MoE) architectures to further enhance sample efficiency and task generalization.

IV. Discussion

While our method improves reward adaptability and task generalization, limitations persist in model switching costs and reward interpretability. Future efforts may include structured prompting [41], [42], hybrid reward learning [43], and MoE architecture integration.

V. Conclusion

We propose a dual-dynamic optimization framework for CoT-based RL reward generation, incorporating temperature regulation and model selection. Our results demonstrate improved convergence, stability, and reward quality across tasks, laying a foundation for scalable, self-adaptive reward engineering.

References

[1] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[2] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[3] J. Skalse and et al., "Misspecification in inverse reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 73, pp. 517–550, 2022.

[4] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *Advances in neural information processing systems*, vol. 31, 2018.

[5] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," *Advances in neural information processing systems*, vol. 30, 2017.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[7] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[8] M. Nguyen, A. Baker, C. Neo, A. Roush, A. Kirsch, and R. Shwartz-Ziv, "Turning up the heat: Min-p sampling for creative and coherent llm outputs," *arXiv preprint arXiv:2407.01082*, 2024.

[9] M. Peeperkorn, T. Kouwenhoven, D. Brown, and A. Jordanous, "Is temperature the creativity parameter of large language models?" *arXiv preprint arXiv:2405.00492*, 2024.

[10] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2022, pp. 287–311.

[11] N. Du, Y. Li, Z. Dai, N. Shazeer, W. Fedus, M. Tan, O. Vinyals, Q. Le, J. Dean, Z. Chen *et al.*, "Glam: Efficient scaling of language models with mixture-of-experts," in *International Conference on Machine Learning*. PMLR, 2022, pp. 5712–5721.

[12] S. Huang, L. Yang, Y. Song, S. Chen, L. Cui, Z. Wan, Q. Zeng, Y. Wen, K. Shao, W. Zhang *et al.*, "Thinkbench: Dynamic out-of-distribution evaluation for robust llm reasoning," *arXiv preprint arXiv:2502.16268*, 2025.

[13] Z. Qi, H. Luo, X. Huang, Z. Zhao, Y. Jiang, X. Fan, H. Lakkaraju, and J. Glass, "Quantifying generalization complexity for large language models," *arXiv preprint arXiv:2410.01769*, 2024.

[14] W. Lu, X. Zhao, J. Spisak, J. H. Lee, and S. Wermter, "Mental modeling of reinforcement learning agents by language models," *arXiv preprint arXiv:2406.18505*, 2024.

[15] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.

[16] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg, "Intrinsically motivated reinforcement learning: An evolutionary perspective," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, pp. 70–82, 2010.

[17] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[18] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[19] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Automated dense reward function generation for reinforcement learning," *arXiv preprint arXiv:2309.11489*, 2023.

[20] S. Sun, R. Liu, J. Lyu, J.-W. Yang, L. Zhang, and X. Li, "A large language model-driven reward design framework via dynamic feedback for reinforcement learning," *arXiv preprint arXiv:2410.14660*, 2024.

[21] Z. Xie, J. Gao, L. Li, Z. Li, Q. Liu, and L. Kong, "Jailbreaking as a reward misspecification problem," *arXiv preprint arXiv:2406.14393*, 2024.

[22] Y. Wu, Z. Sun, H. Yuan, K. Ji, Y. Yang, and Q. Gu, "Self-play preference optimization for language model alignment," *arXiv preprint arXiv:2405.00675*, 2024.

[23] M. Song, Z. Su, X. Qu, J. Zhou, and Y. Cheng, "Prmbench: A fine-grained and challenging benchmark for process-level reward models," *arXiv preprint arXiv:2501.03124*, 2025.

[24] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[25] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[26] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[27] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint*, 2023.

[28] K. Shum, S. Diao, and T. Zhang, "Automatic prompt augmentation and selection with chain-of-thought from labeled data," *arXiv preprint arXiv:2302.12822*, 2023.

[29] I.-C. Baek, S.-H. Park, S. Earle, Z. Jiang, N. Jin-Ha, J. Togelius, and K.-J. Kim, "Pcgrllm: Large language model-driven reward design for procedural content generation reinforcement learning," *arXiv preprint arXiv:2502.10906*, 2024. [Online]. Available: https://arxiv.org/abs/2502.10906

[30] X. Zhu, J. Du, Q. Fu, and L. Chen, "Llm-based reward engineering for reinforcement learning: A chain of thought approach," in *2025 10th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*. IEEE, 2025, pp. 222–227.

[31] Y. Zhu, J. Li, G. Li, Y. Zhao, Z. Jin, and H. Mei, "Hot or cold? adaptive temperature sampling for code generation with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 437–445.

[32] S. Zhang, Y. Bao, and S. Huang, "Edt: Improving large language models' generation by entropy-based dynamic temperature sampling," *arXiv preprint arXiv:2403.14541*, 2024.

[33] N. Cecere, A. Bacciu, I. F. Tobías, and A. Mantrach, "Monte carlo temperature: a robust sampling strategy for llm's uncertainty quantification methods," *arXiv preprint arXiv:2502.18389*, 2025.

[34] C.-C. Chang, D. Reitter, R. Aksitov, and Y.-H. Sung, "Kl-divergence guided temperature sampling," *arXiv preprint arXiv:2306.01286*, 2023.

[35] E. Evstafev, "The paradox of stochasticity: Limited creativity and computational decoupling in temperature-varied llm outputs of structured fictional data," *arXiv preprint arXiv:2502.08515*, 2025.

[36] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Q. V. Le, J. Laudon *et al.*, "Mixture-of-experts with expert choice routing," *Advances in Neural Information Processing Systems*, vol. 35, pp. 7103–7114, 2022.

[37] Y. Li, "Llm bandit: Cost-efficient llm generation via preference-conditioned dynamic routing," *arXiv preprint arXiv:2502.02743*, 2025.

[38] J. Hu, Y. Wang, S. Zhang, K. Zhou, G. Chen, Y. Hu, B. Xiao, and M. Tan, "Dynamic ensemble reasoning for llm experts," *arXiv preprint arXiv:2412.07448*, 2024.

[39] K. Nakaishi, Y. Nishikawa, and K. Hukushima, "Critical phase transition in large language models," *arXiv preprint arXiv:2406.05335*, 2024.

[40] Y. Li, J. Zhang, S. Feng, P. Yuan, X. Wang, J. Shi, Y. Zhang, C. Tan, B. Pan, Y. Hu *et al.*, "Revisiting self-consistency from dynamic distributional alignment perspective on answer aggregation," *arXiv preprint arXiv:2502.19830*, 2025.

[41] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *arXiv preprint arXiv:2211.12588*, 2022.

[42] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "Pal: Program-aided language models," pp. 10 764–10 799, 2023.

[43] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9460–9471, 2022.