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

Xinning Zhu[a], Jinxin Du[a], Longfei Huang[a], Lunde Chen[a,*]

[a]*Sino-European School of Technology, Shanghai University, Shanghai, China*

Abstract

Designing effective reward functions remains a challenge in applying reinforcement learning (RL) to real-world tasks. This paper proposes DyCoT-RE, a large language model (LLM)-driven reward engineering framework that integrates Chain-of-Thought (CoT) reasoningg with a dual-dynamic optimization strategy to automate and enhance reward function design. The framework uses structured CoT reasoning iteratively generate and refine structured reward code. It further incorporates a dual-dynamic optimization mechanism, comprising a temperature adjustment module that modulates the sampling temperature based on policy entropy trends, and a model switching module that dispatches language models with different capabilities to produce distinct reward components. Evaluations across four RL environments: CartPole, BipedalWalker, Ant, and SpaceMining. The latter is a novel environment, absent from standard RL benchmarks and LLMs pretraining corpora. DyCoT-RE consistently achieves higher average rewards and faster learning compared to human-designed and non-CoT baselines, as well as single-optimization variants. These results demonstrate its potential as a scalable and adaptive solution for complex, evolving reward design.

Keywords: Reinforcement learning, reward engineering, large language models, chain-of-thought reasoning, dynamic temperature adjustment, model selection

*Corresponding author
Email: lundechen@shu.edu.cn

1. Introduction

Reinforcement learning has achieved impressive results across diverse domains. Indeed, 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 suggest that although reinforcement learning offers strong theoretical flexibility, its real-world applications are frequently limited by the complexity and domain-specific knowledge required to design effective reward functions. [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 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 ensuring closer correspondence to intended objectives. This structured reasoning process suggests a promising direction for reward engineering, as it provides a means of translating task descriptions into reward functions with well-defined objectives.

However, existing CoT-based reward generation approaches typically use static sampling parameters and fixed model configurations, which may limit their adaptability. Recent studies have emphasized the need for dynamic adaptation in LLM-based systems to improve sample efficiency and maintain adaptability to evolving task requirements. 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. Collectively, these findings demonstrate that dynamic mechanisms such as temperature adjustment and expert model selection can enhance adaptability in LLM-driven systems.

In this work, we propose DyCoT-RE, a reward engineering framework that combines 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 intended task objectives. The dual-dynamic optimization strategy employs two mechanisms to support this process: entropy-guided temperature adjustment balances exploration and exploitation, while the dynamic model selection module routes sub-tasks to specialized LLMs based on performance feedback. By coupling these components within a closed-loop evolutionary search process, DyCoT-RE facilitates systematic reward engineering and improves training efficiency in complex reinforcement learning tasks.

We evaluate DyCoT-RE on the standard RL environments CartPole, Bipedal-Walker, and Ant to assess its performance across different control tasks. While these environments are widely used, recent studies have raised concerns that LLMs may possess latent knowledge about them from pretraining corpora, potentially leading to prompt leakage and evaluation bias[12, 13, 14]. To address this concern and test DyCoT-RE's adaptability to unfamiliar scenarios, we additionally introduce a custom-designed environment, SpaceMining[1], which represents a novel task domain. Experimental results show that DyCoT-RE consistently achieves higher average rewards and faster convergence than baseline and non-CoT methods across all evaluated environments.

The remainder of this paper is organized as follows. Section 2 reviews

[1]Project website: `https://reveurmichael.github.io/space_mining/`.

related work in reward engineering, chain-of-thought reasoning, and dynamic optimization in LLM-based systems. Section 3 introduces the DyCoT-RE framework, including CoT-based reward construction and the dual-dynamic optimization strategy. Section 4 presents experimental evaluations, including setup, benchmark comparisons, ablations, and evaluation in a novel environment. Section 5 discusses the main findings, limitations, and potential directions for future work. Section 6 concludes the paper.

2. Related Work

2.1. 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 studies have increasingly leveraged LLMs for automated reward engineering in reinforcement learning. Early demonstrations showed that language descriptions can serve as reward signals or be translated into executable reward code, notably Reward Design with Language Models [18] and Language-to-Rewards (L2R) [19]. Building on this concept, rapid progress has been made, from simple prompts to automatic code generation and closed-loop optimization. Representative works include Text2Reward [20] for dense reward shaping with iterative refinement, VLM-based zero-shot rewards [21] using vision-language models, and LLM4PG [22] for automated preference labeling.

Among these, Eureka [23] demonstrates a notable automated framework that treats LLMs as autonomous agents, iteratively generating, mutating, and selecting reward code via performance-based tournaments. In their evaluation, Eureka achieved performance improvements over expert-designed rewards across several RL tasks, with reported gains of approximately 52% in multiple tested environments. CARD [24] further advanced this paradigm with dynamic feedback pipelines that integrate process, trajectory, and preference evaluation for automated reward optimization. Recent developments explore progress-function rewards [25] and self-judging mechanisms [26], indicating a shift toward closed-loop, feedback-driven approaches that incorporate language reasoning with optimization strategies.

Building on these developments, future research can explore how LLMs can more effectively bridge human intent and reinforcement learning objectives, leverage feedback for reward generation, and address increasingly complex real-world tasks.

2.2. 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. [27] demonstrated such prompts significantly improve performance in arithmetic and commonsense tasks. Building upon this, few-shot CoT [28] introduced demonstrations of stepwise solutions to guide model reasoning, while self-consistency decoding [29] aggregated multiple sampled reasoning paths to enhance answer robustness.

Further developments have demonstrated the effectiveness of CoT reasoning in language models. DeepSeek [30] showed that structured reasoning capabilities can emerge through self-evolution mechanisms, achieving complex CoT reasoning without supervised fine-tuning. Automatic prompt optimization methods [31] reduce manual engineering efforts by refining prompts based on data-driven insights.

Recent work such as PCGRLLM [32] explored CoT-based LLM reward design for procedural content generation in RL, demonstrating feasibility in structured game environments. In parallel, Zhu et al. [33] proposed a CoT-based reward engineering approach that translates natural language task descriptions into RL reward functions using LLMs, validating its effectiveness in a standard environment. 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 demonstrated effectiveness in improving LLM interpretability and reasoning performance, its application to automated reward engineering in RL remains underexplored. Integrating CoT reasoning with dynamic optimization strategies may offer potential benefits for developing more adaptive reward generation systems.

2.3. Dynamic Temperature Adjustment and Model Selection

Dynamic temperature adjustment and model selection have emerged as important optimization strategies for improving LLM-based system performance. Temperature, as a sampling hyperparameter, modulates the randomness of LLM outputs, influencing the balance between exploration and exploitation.

Recent studies have explored adaptive temperature mechanisms from multiple angles. Zhu et al. [34] proposed AdapT, which adjusts decoding temperature based on token-level difficulty in code generation tasks. In parallel, Zhang et al. [35] developed Entropy-based Dynamic Temperature (EDT) sampling to regulate output diversity in natural language generation. Taking a different approach, Cecere et al. [36] focused on uncertainty quantification, introducing Monte Carlo Temperature for robust sampling under distribution shifts. Chang et al. [37] employed KL-divergence to guide temperature control for adaptive exploration.

In the realm of model selection, researchers have pursued efficiency and robustness through various mechanisms. Zhou et al. [40] enhanced Mixture-of-Experts efficiency with expert choice routing, whereas Li et al. [41] developed preference-conditioned dynamic routing for cost-effective generation. Hu et al. [42] integrated outputs from multiple specialized LLMs through Dynamic Ensemble Reasoning. Li et al. [43] reconsidered self-consistency decoding from a distributional alignment perspective, contributing to our understanding of expert aggregation strategies.

Despite recent advances, temperature adaptation has mainly aimed at enhancing generative diversity and calibration, while model selection focuses on efficiency and specialization. Extending these techniques to reward generation offers a promising direction for reinforcement learning.

3. Methodology

This section details the implementation of DyCoT-RE, focusing on the CoT-based reward construction process and the dual-dynamic optimization

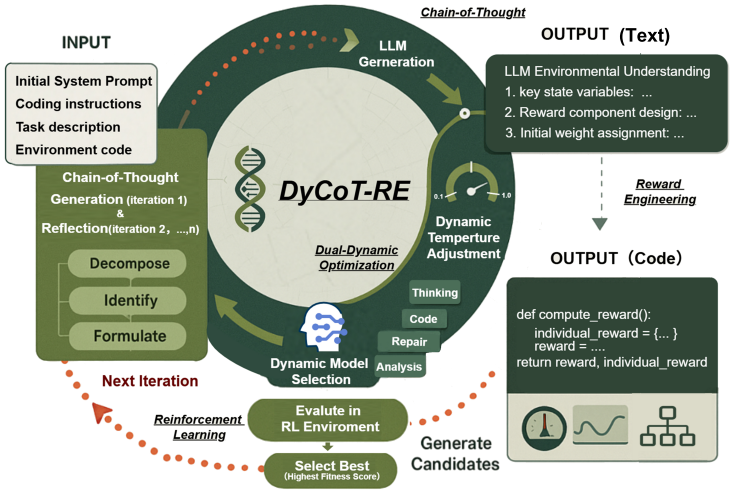

Figure 1: DyCoT-RE framework integrating CoT reasoning, dynamic temperature adjustment, and model selection in an evolutionary optimization loop.

mechanisms.

3.1. Framework Overview

Figure 1 presents an overview of DyCoT-RE, which comprises three key components: structured CoT reward decomposition, dynamic temperature adjustment, and dynamic model selection. The framework receives four types of inputs: natural language task descriptions, environment interfaces, system-level prompts, and implementation instructions.

These inputs are processed through a pipeline involving four expert LLMs, referred to as Think LLM, Code LLM, Repair LLM, and Analysis LLM. These models, denoted respectively as $\mathcal{M}$think, $\mathcal{M}$code, $\mathcal{M}$repair, and $\mathcal{M}$analysis are dynamically assigned by a model selection module to handle distinct aspects of reward generation through structured CoT reasoning. Meanwhile, a dynamic temperature adjustment module modulates the sampling parameters of these LLMs based on performance feedback.

Through this coordinated process, DyCoT-RE produces structured reward

functions of the form:

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

where $r_i(s,a)$ denotes the i-th reward component and w_i is its corresponding weight. These components are derived through structured CoT reasoning and reflect interpretable sub-goals. The resulting reward function is deployed in the RL environment, and its performance guides the next iteration of temperature tuning and model routing.

The following subsections detail each component's implementation and the coordination mechanisms between them.

3.2. Chain-of-Thought Reasoning for Reward Engineering

The CoT reasoning module decomposes task description d into interpretable reward components. Each component $r_i(s,a)$ is generated through structured reasoning:

$$r_i(s,a) = \mathcal{M}_{\text{code}}(\text{CoT}_{\mathcal{M}_{\text{think}}}(S, C, D, E)), \tag{2}$$

where S represents the initial system instruction, C the coding instructions, D the task description, and E the environment code. The $\mathcal{M}_{\text{think}}$ processes these inputs through CoT reasoning to identify subgoals and constraints for component i, then $\mathcal{M}_{\text{code}}$ translates this structured analysis into executable reward functions.

The complete reward function follows Eq. (1), where weights w_i reflect subgoal priorities inferred during the CoT process. To evaluate performance independently of the generation process, we compute a fitness score alongside component-level metrics during training. The $\mathcal{M}_{\text{analysis}}$ examines these metrics to iteratively refine both $r_i(s,a)$ and w_i, optimizing toward:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right]. \tag{3}$$

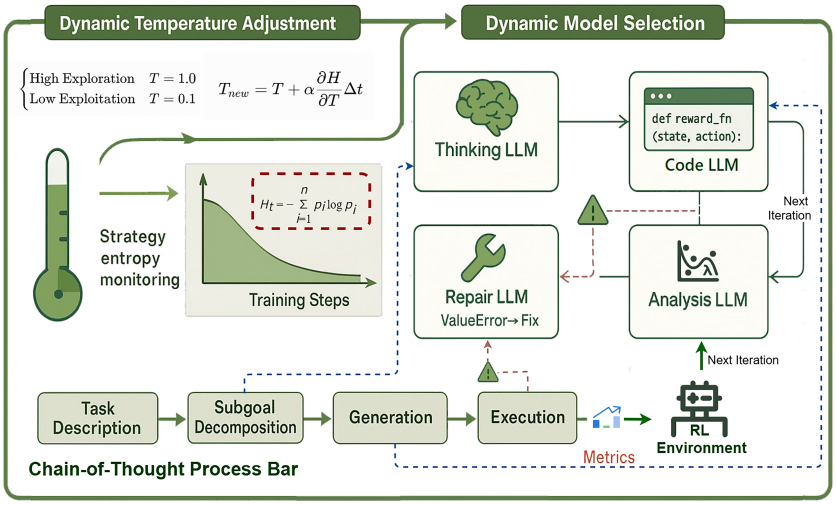

Figure 2: DyCoT-RE control flow integrating temperature modulation and model selection within CoT iterative reasoning.

3.3. Dual-Dynamic Optimization Strategy

Figure 2 illustrates the coupled feedback architecture of DyCoT-RE, where temperature adjustment and model selection operate in synergy to enhance reward function generation and RL performance.

3.3.1. Dynamic Temperature Adjustment

Temperature adjustment modulates the sampling temperature T based on policy entropy H, confidence C, and performance R. Entropy reflects generative diversity, confidence measures output stability, and performance evaluates reward improvement relative to historical best.

The update rule is formulated as:

$$T_{k+1} = \text{clip}\left(T_k + \alpha \frac{\partial T}{\partial H_k} \Delta t\right), \tag{4}$$

where α is a learning rate, and the gradient $\frac{\partial T}{\partial H_k}$ reflects entropy-driven adjustment. An alternative implementation combines smoothing and multiplicative

adjustment:

$$T_{k+1} = \text{clip}\left(\alpha T_k + (1-\alpha) T_k f(H_k, C_k, R_k)\right). \tag{5}$$

Here, $f(H, C, R)$ integrates:

$$f(H, C, R) = f_R(R) f_H(H) f_C(C), \tag{6}$$

where f_R ensures performance protection, f_H regulates entropy bounds, and f_C maintains confidence stability.

3.3.2. Dynamic Model Selection

Dynamic model selection adaptively routes sub-tasks to specialized LLMs, leveraging their complementary strengths across the reward engineering pipeline.

The four LLM classes include: $\mathcal{M}_{\text{think}}$ for task understanding and semantic decomposition, $\mathcal{M}_{\text{code}}$ for reward function synthesis, $\mathcal{M}_{\text{repair}}$ for code correction, and $\mathcal{M}_{\text{analysis}}$ for performance evaluation and sub-reward weight updates.

Formally, the decomposition and generation process can be expressed as:

$$g_i = \mathcal{M}_{\text{think}}(S, C, D, E), \tag{7}$$

$$r_i(s, a) = \mathcal{M}_{\text{code}}(g_i, C), \tag{8}$$

$$w_i^{(t+1)} = \mathcal{M}_{\text{analysis}}(w_i^{(t)}, \text{fitness}^{(t)}, \{r_j^{(t)}\}), \tag{9}$$

Repair LLM intervenes when $\mathcal{M}_{\text{code}}$ outputs execution errors during evaluation.

Model selection at iteration k follows:

$$M_{k+1} = \begin{cases} \arg\max_{m \in \mathcal{M}_s} \text{Perf}(m), & 1-\epsilon, \\ \text{Random}(\mathcal{M}_s \setminus \{M_k\}), & \epsilon, \end{cases} \tag{10}$$

where $\mathcal{M}_s$ is the model pool for stage s, Perf(m) the performance score, and ϵ the exploration rate ensuring selection diversity.

At each iteration, the next model M_{k+1} is selected from pool $\mathcal{M}_s$ using epsilon-greedy selection, where the highest-performing model is chosen with probability $1 - \epsilon$ and a random model with probability ϵ. The pool includes a Repair LLM for error correction during reward code evaluation.

3.3.3. Joint Adaptive Optimization

Temperature adjustment and model selection operate as coordinated mechanisms within the optimization loop. Temperature τ influences the sampling distribution of reward candidates:

$$r_i(s, a) \sim p(r_i|g_i, \tau), \tag{11}$$

where g_i represents the subgoal specification from $\mathcal{M}_{\text{think}}$. This sampling process modulates exploration patterns in the reward space.

The joint objective combines both mechanisms:

$$(\tau^*, M^*) = \arg\max_{\tau, M} \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{H} \gamma^t R_{\tau,M}(s_t, a_t) \right], \tag{12}$$

where $R_{\tau,M}(s, a)$ denotes the reward function generated under temperature τ and model configuration M, and H represents the episode horizon.

4. Experiments

4.1. Experimental Setup

DyCoT-RE is evaluated on four reinforcement learning environments, which span diverse task difficulties, action spaces, and complexity levels. Standard benchmarks include CartPole, BipedalWalker, and Ant, covering discrete control tasks and high-dimensional continuous locomotion.

To assess performance on novel environments beyond pretrained knowledge, we introduce SpaceMining, a custom environment with multi-objective resource collection and dynamic energy constraints. Unlike standard environments that

LLMs may encounter during pretraining, SpaceMining tests whether reward design can succeed through task understanding rather than prior knowledge.

4.1.1. Baselines and Comparison Strategies

We evaluate DyCoT-RE against two primary baselines:

(1) Human-designed rewards: Manually crafted reward functions from standard Gymnasium[44] implementations for CartPole, BipedalWalker, and Ant. For SpaceMining, we design corresponding reward functions following similar principles.

(2) Eureka: An established LLM-based framework for automated reward synthesis that employs iterative code generation and performance-based selection. We adapt its pipeline to Gymnasium-based evaluations for fair comparison.

4.1.2. Ablation Comparisons

To evaluate individual contributions within DyCoT-RE, we include the following controlled variants:

- Zero-shot reward generation: Reward generation using direct prompts without CoT reasoning.

- Static temperature: Sampling temperature remains fixed throughout training.

- Single model: A single LLM handles all reward generation tasks without dynamic model selection.

These comparisons isolate the contributions of CoT-based reasoning and the dual-dynamic mechanisms.

4.1.3. Implementation Details

We evaluate DyCoT-RE on three environments from the Gymnasium suite — CartPole-v1[2], BipedalWalker-v3[3], and Ant-v5[4]—alongside a custom-designed SpaceMining environment for evaluating performance in unseen domains.

[2] `https://gymnasium.farama.org/environments/classic_control/cart_pole/`
[3] `https://gymnasium.farama.org/environments/box2d/bipedal_walker/`
[4] `https://gymnasium.farama.org/environments/mujoco/ant/`

All experiments use Proximal Policy Optimization (PPO) from Stable-Baselines3 with Adam optimizer, learning rate of 3e-4, batch size 64, GAE parameter $\lambda = 0.95$, and discount factor $\gamma = 0.999$.

DyCoT-RE generates reward functions using local LLMs served via Ollama, sampling eight reward candidates in parallel at each CoT iteration.

Final performance metrics are computed as the mean over ten test episodes with different random seeds.

Table 1: Task specifications for evaluated RL environments.

Environment	State/Action Dims	Episode Steps
CartPole-v1	4 / 1 discrete	500
BipedalWalker-v3	24 / 4 continuous	1600
Ant-v5	111 / 8 continuous	1600
SpaceMining	53 / 3 continuous	1200

4.2. Performance Across Environments

We compare DyCoT-RE against Human-designed and Eureka baselines across all evaluated environments. Figure 3 shows that DyCoT-RE achieves higher average rewards in all tested scenarios, with performance gaps increasing as task complexity grows.

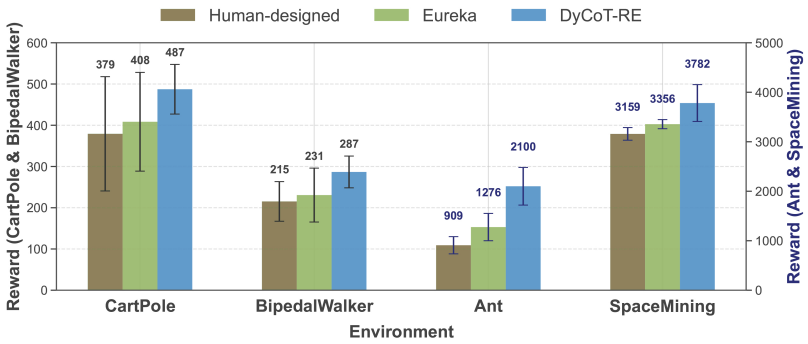

Figure 3: Final average rewards across environments: DyCoT-RE (blue), Eureka (light grass), and Human-designed (dark grass).

Figure 4 provides insights into the learning dynamics. In CartPole, all

methods converge quickly due to low-dimensional discrete dynamics, though DyCoT-RE converges slightly faster convergence. In BipedalWalker, DyCoT-RE exhibits a two-phase pattern: initial exploration volatility followed by stable growth and smoother long-term convergence. In the Ant environment, all methods initially encounter performance valleys, but DyCoT-RE recovers more quickly and achieves sustained improvement to higher fitness scores, while Eureka shows instability throughout training. In SpaceMining, DyCoT-RE maintains steady progress while baselines show flat or unstable performance, demonstrating effective generalization to novel tasks.

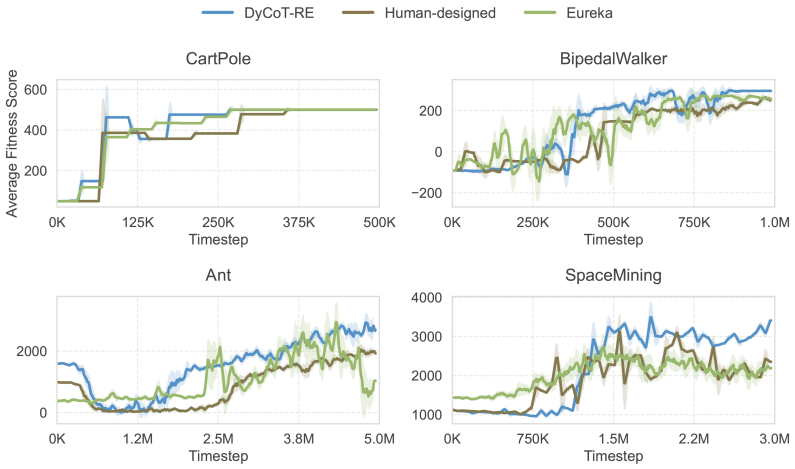

Figure 4: Learning curves across environments. DyCoT-RE exhibits consistent advantages in stability and long-term reward growth, especially in complex settings.

Table 2 compares performance across four tasks using three metrics: Sample Efficiency measures the number of timesteps required to reach a threshold reward, defined as 80% of the task-specific maximum score. Success Rate indicates the proportion of runs that eventually exceed this threshold. Late Gain quantifies the reward improvement from the early performance plateau to the final evaluation, reflecting long-term optimization potential.

DyCoT-RE outperforms baselines across all evaluated metrics: it reaches target performance faster, succeeds more reliably, and continues improving in

Table 2: Cross-task performance comparison. Blue : the best value per column, green : the second-best. Success Rate is omitted for Human due to deterministic reward completion.

Task	Method	Sample Eff. ↓	Succ. Rate ↑	Late Gain ↑
CartPole	DyCoT-RE	75k	98%	450
	Eureka	115k	95%	440
	Human	286k	–	400
BipedalWalker	DyCoT-RE	393k	92%	387
	Eureka	428k	83%	366
	Human	720k	–	309
Ant	DyCoT-RE	16.19M	85%	5398
	Eureka	21.53M	70%	2412
	Human	19.92M	–	-240
SpaceMining	DyCoT-RE	1.31M	80%	2285
	Eureka	1.74M	65%	189
	Human	1.56M	–	1230

later training stages. These advantages are most evident in high-dimensional environments, where static reward design approaches show limited adaptability.

4.3. Ablation Study

This section conducts an ablation study to examine the individual contribution of each component in DyCoT-RE, including CoT reasoning, temperature adjustment, and model selection, as well as their combined synergistic effects.

4.3.1. Impact of CoT Reasoning

We evaluate the impact of incorporating CoT reasoning into reward engineering by comparing DyCoT-RE's CoT-enabled variant against a zero-shot baseline lacking structured intermediate reasoning.

Figure 5 summarizes reward distributions for both methods across the four benchmark environments. A consistent pattern emerges: CoT-enabled rewards exhibit higher means, lower variance, and reduced outlier incidence compared to zero-shot. In CartPole and SpaceMining, CoT-enabled results are tightly clustered near the environment's upper performance bounds, whereas zero-shot

results are widely dispersed, with a substantial proportion of runs yielding sub-optimal or even near-zero rewards. This reflects CoT's ability to systematically decompose task objectives into interpretable sub-rewards, enhancing sample efficiency and stabilizing policy learning.

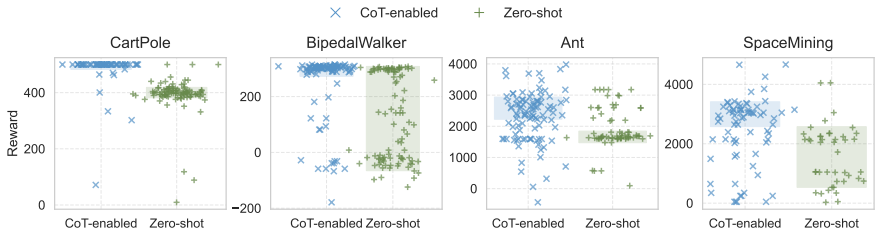

Figure 5: Final reward distributions for CoT-enabled vs zero-shot reward generation across environments.

In BipedalWalker, a notable deviation is observed. CoT-enabled runs yield a concentrated reward range between 250 and 310, with virtually no severely negative outcomes. In contrast, the zero-shot condition presents a clear bimodal distribution: one cluster achieves moderate reward levels around 150 to 250, while the other collapses into strongly negative regimes. This instability reflects the inability of zero-shot reward generation to encode essential gait balance or locomotion constraints, often resulting in policy failure.

In Ant, zero-shot results are narrowly concentrated near baseline values associated with default movement penalties. This indicates that while these reward functions do not collapse, they lack task-driving gradients for effective behavior shaping. In contrast, CoT-based rewards achieve a wider spread toward higher values, reflecting meaningful progress and more informative optimization signals.

Taken together, these results illustrate that CoT reasoning reliably improves reward interpretability and training stability, especially in settings with complex objectives.

4.3.2. Impact of Temperature Adjustment

Due to computational constraints and consistent observed trends across environments, temperature adjustment and model selection ablation were conducted on BipedalWalker as a representative continuous control task, while CoT reasoning was evaluated on all four environments to confirm its general effectiveness.

This section investigates the effect of temperature settings by sweeping $T \in [0.0, 1.0]$ on the BipedalWalker environment.

Table 3 reports BipedalWalker results under different temperature settings, including average and maximum fitness, standard deviation, and reward efficiency ratio. DyCoT-RE achieves the best performance across all metrics, with a large margin in efficiency ratio due to its high average fitness and low variability. Among fixed-temperature settings, moderate values (0.4–0.6) deliver relatively strong results, yet still fall short of DyCoT-RE.

Table 3: BipedalWalker performance under different temperature settings. Blue : the best value per column, green : the second-best, green : the third-best

Temp	Avg Fit↑	Max Fit ↑	Std↓	Eff. Ratio↑
0.0	81.93	301.15	148.67	0.55
0.1	149.56	310.94	155.22	0.96
0.2	242.74	313.60	125.47	1.93
0.3	210.07	311.17	140.49	1.50
0.4	244.19	311.94	107.02	2.28
0.5	215.84	310.69	131.01	1.65
0.6	248.11	312.48	120.45	2.06
0.7	225.88	312.42	131.43	1.72
0.8	74.33	310.42	143.91	0.52
0.9	145.02	310.02	148.51	0.98
1.0	192.38	310.40	138.37	1.39
DyCoT-RE	292.8	315.6/87.2	55.2	5.30

Static sweeps show that mid-range temperatures yield higher fitness and lower variance than extremes. DyCoT-RE's dynamic temperature regulation consistently outperforms these static settings, adapting temperature during training to maintain an effective exploration-exploitation trade-off.

4.3.3. Impact of Model Selection

Finally, we evaluate the impact of DyCoT-RE's dynamic model selection by comparing it against individual static LLM baselines on the BipedalWalker environment. Table 4 compares different LLM backends on BipedalWalker in terms of average fitness, peak performance, stability, and reward efficiency.

DyCoT-RE achieves the highest average fitness with exceptionally low standard deviation and the best efficiency ratio, demonstrating its capability to consistently generate high-quality rewards across diverse sub-tasks.

In contrast, while codegemma achieves competitive performance, its higher standard deviation results in a substantially lower efficiency ratio, indicating reduced stability and generalizability compared to DyCoT-RE.

Table 4: Performance comparison of different LLM backends (N=20) in BipedalWalker. Blue : the best value per column, green : the second-best

Model (size)	Avg Fit ↑	Max Fit ↑	Std ↓	Eff. Ratio ↑
codegemma (7b)	260.8	301.8	34.2	3.46
codeqwen (7b)	32.6	307.6	98.42	0.33
deepseek-coder (6.7b)	-4.5	295.3	98.19	-0.05
deepseek-r1 (8b)	44.6	321.2	122.07	0.37
gemma3 (4b)	278.0	314.8	142.04	3.39
llama3.1 (8b)	162.2	318.4	150.75	1.08
qwen2.5 (7b)	177.0	319.5	151.51	1.17
qwen2.5-coder (7b)	125.6	313.7	163.16	0.77
DyCoT-RE	297.28	316.57	89.47	6.46

While individual models occasionally excel on specific tasks, DyCoT-RE's dynamic model routing achieves broadly consistent performance, balancing reward quality and stability in complex continuous control environments.

In summary, the ablation study indicates that each module—CoT reasoning, temperature adjustment, and model selection—plays a substantive role in enhancing DyCoT-RE's performance.

4.4. DyCoT-RE in SpaceMining: Results in an LLM-Unseen Environment

To evaluate DyCoT-RE in an out-of-distribution scenario without task-specific priors, we constructed the SpaceMining environment—a multi-objective domain

involving dynamic hazards and energy constraints. This setting challenges the model to formulate effective rewards from natural language alone, testing its semantic generalization and reasoning capabilities.

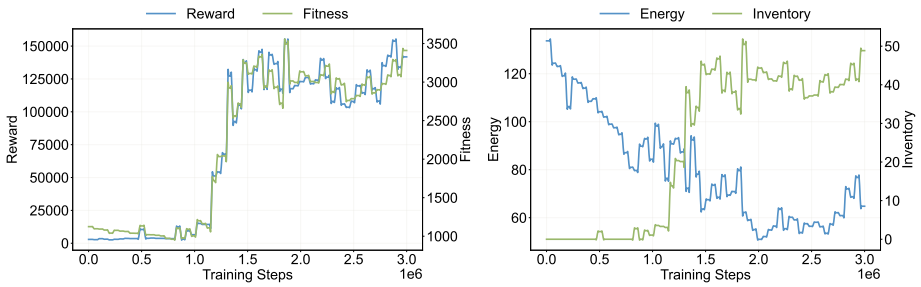

Figure 6: Training dynamics in SpaceMining. Left: reward and fitness improve steadily, showing effective long-term optimization. Right: stable energy levels and growing inventory indicate successful energy-resource balancing.

Figure 6 illustrates the learning trajectory of DyCoT-RE. From early-stage exploration to late-stage optimization, we observe concurrent increases in reward and fitness, with consistent energy regulation and inventory accumulation. These trends reflect reward structures that incentivize planning, efficiency, and safe behavior.

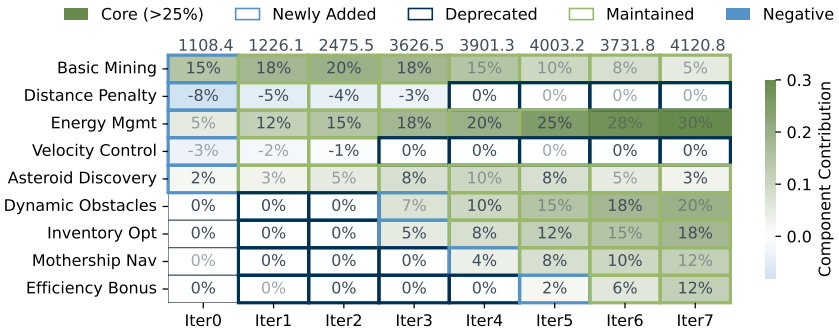

Figure 7: The evolution of normalized reward component contributions in DyCoT-RE for SpaceMining

Figure 7 shows the relative contribution of each reward component during

training iterations. Positive and negative values indicate reinforcement and penalty signals respectively, while zero values with reduced opacity show inactive components. Border colours denote component lifecycle stages including new, maintained, core, and deprecated components, with column labels showing corresponding fitness scores. Component contribution is computed as $\text{Contr}_i = \frac{\overline{r_i}}{\sum_j \overline{r_j}} \times 100\%$, where $\overline{r_i}$ is the average reward from component i.

The system begins with a minimal setup focused on mining progress, distance penalties, and energy tracking. As training proceeds, DyCoT introduces new components such as obstacle avoidance in Iteration 3, inventory-based delivery efficiency in Iteration 5, and asteroid discovery incentives in Iteration 7. These additions are not arbitrary but driven by observed performance shifts. For instance, after dynamic obstacle handling is introduced, the agent's survival rate improves significantly. Conversely, some components such as velocity control lose their relevance and are phased out by Iteration 3, as their function becomes redundant once obstacle-aware navigation is in place. The energy management module, on the other hand, steadily increases in importance and becomes a core contributor throughout the later iterations. This dynamic component lifecycle—characterized by incremental expansion, adaptive weighting, and timely deprecation—demonstrates DyCoT's capacity for modular, interpretable, and performance-aligned reward optimization.

Table 5: DyCoT Strategy Metrics Across Iterative Architectures. Blue : the best value per column, green : the second-best.

Iter	Temperature	Entropy	Confidence	Best Reward	Reward Gain (%)
0	0.800	0.912	0.642	880.2	–
1	0.700	0.946	0.705	1124.4	+27.7
2	0.669	0.928	0.748	1320.3	+17.4
3	0.639	0.986	0.778	3475.5	+163.2
4	0.600	0.934	0.791	3551.0	+2.2
5	0.556	0.914	0.807	3801.3	+7.0
6	0.534	0.962	0.815	3876.2	+2.0
7	0.517	0.972	0.824	4003.2	+3.3

Table 5 illustrates the evolution of key strategy-level metrics throughout all

DyCoT iterations. As the architecture matures from Iteration 0 to 7, we observe a consistent decline in sampling temperature, indicating increased confidence in policy generation. Entropy fluctuates with architecture changes, notably peaking at Iteration 3 when major reward components are introduced. Confidence improves steadily, rising from 0.64 to 0.82, in parallel with fitness gains. Reward gain is computed relative to the previous iteration, with Iteration 3 marking a significant performance inflection point. Subsequent gains are more incremental, reflecting architectural stabilization.

In summary, DyCoT-RE exhibits the capacity for iterative reward adaptation. It evolves from simple distance-based signals to structured reward components that support coordinated behaviors such as exploration, energy regulation, and goal-oriented progression—achieving consistent performance in environments with minimal prior knowledge.

5. Discussion

This work presents DyCoT-RE, a reward engineering framework that integrates CoT reasoning with dual-dynamic optimization strategies for automated reward function design. Our experimental evaluation across four RL environments demonstrates consistent performance improvements over human-designed rewards and existing automated baselines, with the most significant gains observed in complex, multi-objective environments.

Three findings emerge from our evaluation. First, CoT decomposition improves reward interpretability and stability by breaking complex objectives into manageable components. Second, dynamic strategies outperform static configurations: adaptive temperature control balances exploration-exploitation better than fixed settings, while model selection leverages diverse LLM strengths for higher performance and consistency. Third, SpaceMining results demonstrate DyCoT-RE's ability to generate effective reward functions even in unfamiliar environments, suggesting that structured reasoning can compensate for the absence of domain-specific priors.

DyCoT-RE introduces computational overhead and depends on underlying LLM capabilities. Evaluation is limited to four environments, and the framework uses general-purpose models not optimized for reward engineering. Future research could explore specialized models fine-tuned for reward engineering, incorporate multi-modal inputs beyond natural language, expand evaluation to broader environment categories, and develop transfer learning mechanisms to leverage successful reward designs across similar domains.

DyCoT-RE demonstrates that combining structured reasoning with adaptive optimization offers a promising direction for scalable automated reward engineering, particularly in complex RL domains where traditional approaches prove insufficient.

6. Conclusion

This paper introduces DyCoT-RE, a framework that addresses the challenge of automated reward function design in RL. The main contribution lies in combining structured reasoning through CoT prompting with adaptive optimization mechanisms that dynamically adjust both temperature and model selection throughout the reward engineering process.

Evaluation demonstrates that DyCoT-RE consistently produces effective and stable reward functions across tasks ranging from simple control to complex, multi-objective scenarios, including a novel environment unfamiliar to LLMs. The results highlight its potential as a scalable solution for RL tasks requiring structured and evolving reward design for complex objectives.

References

[1] R. S. Sutton, A. G. Barto, et al., Reinforcement learning: An introduction, Vol. 1, MIT press Cambridge, 1998.

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

[3] J. Skalse, et al., Misspecification in inverse reinforcement learning, Journal of Artificial Intelligence Research 73 (2022) 517–550.

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

[5] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, A. Dragan, Inverse reward design, Advances in neural information processing systems 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 33 (2020) 1877–1901.

[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 35 (2022) 27730–27744.

[8] M. Nguyen, A. Baker, C. Neo, A. Roush, A. Kirsch, 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, A. Jordanous, Is temperature the creativity parameter of large language models?, arXiv preprint arXiv:2405.00492 (2024).

[10] W. Fedus, B. Zoph, 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, 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, S. Wermter, Mental modeling of reinforcement learning agents by language models, arXiv preprint arXiv:2406.18505 (2024).

[15] A. Y. Ng, D. Harada, 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, J. Sorg, Intrinsically motivated reinforcement learning: An evolutionary perspective, IEEE Transactions on Autonomous Mental Development 2 (2) (2010) 70–82.

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

[18] M. Kwon, S. M. Xie, K. Bullard, D. Sadigh, Reward design with language models, arXiv preprint arXiv:2303.00001 (2023).

[19] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al., Language to rewards for robotic skill synthesis, arXiv preprint arXiv:2306.08647 (2023).

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

[21] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, D. Lindner, Vision-language models are zero-shot reward models for reinforcement learning, arXiv preprint arXiv:2310.12921 (2023).

[22] Z. Shen, T. Zhu, Q. Sun, S. Gao, J. Li, Beyond human preferences: Exploring reinforcement learning trajectory evaluation and improvement through llms, arXiv preprint arXiv:2406.19644 (2024).

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

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

[25] V. Sarukkai, B. Shacklett, Z. Majercik, K. Bhatia, C. Ré, K. Fatahalian, Automated rewards via llm-generated progress functions (2024). `arXiv:2410.09187`.
URL `https://arxiv.org/abs/2410.09187`

[26] T. Simonds, K. Lopez, A. Yoshiyama, D. Garmier, Rlsr: Reinforcement learning from self reward (2025). `arXiv:2505.08827`.
URL `https://arxiv.org/abs/2505.08827`

[27] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, Advances in neural information processing systems 35 (2022) 22199–22213.

[28] 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 35 (2022) 24824–24837.

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

[30] DeepSeek-AI, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, arXiv preprint (2023).

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

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

[33] X. Zhu, J. Du, Q. Fu, 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.

[34] Y. Zhu, J. Li, G. Li, Y. Zhao, Z. Jin, 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, 2024, pp. 437–445.

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

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

[37] C.-C. Chang, D. Reitter, R. Aksitov, Y.-H. Sung, Kl-divergence guided temperature sampling, arXiv preprint arXiv:2306.01286 (2023).

[38] 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).

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

[40] 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 35 (2022) 7103–7114.

[41] Y. Li, Llm bandit: Cost-efficient llm generation via preference-conditioned dynamic routing, arXiv preprint arXiv:2502.02743 (2025).

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

[43] 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).

[44] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, O. G. Younis, Gymnasium: A standard interface for reinforcement learning environments (2024). arXiv:2407.17032.
URL https://arxiv.org/abs/2407.17032