Regret-Now: A Physics-Inspired Regret Framework for Temporal Knowledge Graph Question Answering with LLMs

Danyu Huang[1], Yao Zhang[2]*, Jun Wang[3], Zhenglu Yang[1]*
[1]Key Laboratory of DISSec, College of Computer Science, Nankai University, Tianjin, China
[2]School of Statistics and Data Science, LPMC, KLMDASR & AAIS, Nankai University, China
[3]College of Mathematics and Statistics Science, Ludong University
danyuhuang@mail.nankai.edu.cn, yaozhang@nankai.edu.cn,
junwang@mail.nankai.edu.cn, yangzl@nankai.edu.cn

Abstract

Large Language Models have achieved impressive results in general reasoning tasks. However, they still face significant challenges when applied to temporal knowledge graph question answering (TKGQA), particularly exhibiting broken temporal reasoning chains and a lack of dynamic error-correction. These limitations hinder their capacity to handle complex temporal logic and make it difficult to recover once a reasoning error occurs. To address this issue, we propose Regret-Now, a novel LLM-based temporal reasoning framework inspired by the physical principle of minimum potential energy. Regret-Now models the reasoning process as a dynamic trajectory moving toward a more stable state, where each step is expected to a lower potential energy. We introduce the Regret Stage that evaluates the "potential energy" of each intermediate reasoning step and triggers real-time rollback if an abnormal rise in potential energy is detected—indicating a likely error. We evaluate Regret-Now on two standard TKGQA benchmarks: CronQuestions and MultiTQ. Experimental results show consistent gains over strong baselines, validating physics-inspired modeling for LLM-based TKGQA. The code can be found at https://github.com/h-yii/Regret-Now.

1 Introduction

Leveraging Large Language Models (LLMs) for TKGQA hinges on their powerful step-by-step reasoning capabilities to navigate Temporal Knowledge Graphs (TKGs) (Liao et al., 2023; Luo et al., 2024; Huang and Chang, 2023). By explicitly modeling time-evolving relations, TKGs provide the structured knowledge and contextual constraints essential for effective temporal reasoning (Liao et al., 2023; Luo et al., 2024). In an ideal scenario, an LLM would follow a coherent reasoning chain to derive a temporally consistent answer, as illustrated in Figure 1 (a) and (b). However, in practice, LLMs often encounter a fundamental bottleneck: **the broken temporal reasoning chain**. They tend to take "reasoning shortcuts"(Chen et al., 2023a), jumping to conclusions without executing all necessary intermediate steps. This omission of critical temporal information, as shown in Figure 1(c), disrupts the entire reasoning process and frequently leads to failure cases.

While some current works attempt to enhance temporal awareness by introducing specialized modules (Mavromatis et al., 2022; Shang et al., 2022), these methods are largely"results-oriented" and lack a mechanism for "real-time monitoring" (Hu et al., 2025) of the reasoning process. Consequently, once the reasoning chain is broken, subsequent steps can only proceed down the erroneous chain. To fundamentally address this issue, we identify two core challenges: 1) Real-time Detection of Reasoning Brokens. The model must be able to acutely detect, at each step, if a decision has deviated from the coherent reasoning chain; and 2) Dynamic Correction of the Reasoning Chain. Upon identifying a break, the model requires a mechanism to proactively "regret" its decision and reconnect the chain.

To address the above challenges, existing works (Yao et al., 2023a; Shinn et al., 2023; Zhou et al., 2025) commonly explore prompting LLM to perform self-correction or introducing another LLM as an evaluator. However, such LLM-dependent methods may fall into subjective biases (Huang et al., 2023), and the interpretability of their evaluation results is also difficult to guaranty. For this reason, we attempt to introduce a quantifiable, external criterion to objectively evaluate the accuracy of each reasoning step. To this end, we draw inspiration from the Principle of Minimum Potential Energy in fundamental physics. This principle states that a physical system always tends towards its stable state of minimum total potential

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 24133–24157

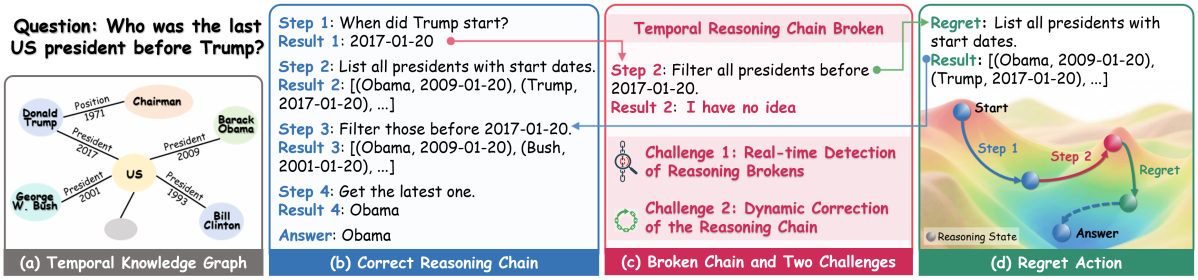


Figure 1: Conceptual illustration of reasoning chains within our Regret-Now framework. (a) and (b) An ideal correct reasoning chain. (c) A case of the broken temporal reasoning chain. (d) A regret operation based on the principle of minimum potential energy.

energy, which provides us with a novel perspective for understanding and optimizing reasoning processes. Based on this principle, we analogize the reasoning process of TKGQA to the movement of a particle in an abstract potential energy field. Herein, the reasoning starting point is the particle, the complete reasoning chain is its trajectory, and the goal is to find the correct answer at the point of minimum potential energy. Therefore, an accurate reasoning step will lead the particle to a lower potential energy state (as in Step 1 in Figure 1 (d)); conversely, a reasoning error is an anomalous "uphill climb" that causes the particle to move away from the answer (as in Step 2 in Figure 1 (d)). By monitoring the potential energy changes during the reasoning process, we are able to identify the broken reasoning chain in real-time and guide the LLM toward the correct direction for the next reasoning step.

Specifically, we propose the **Regret-Now** framework. This framework first constructs a "semantic potential energy field" in which we define "potential energy" as a function of the semantic distance between the reasoning entities. This distance is quantified by their embedding representations. Then, Regret-Now decomposes temporal QA into sequential decision steps, immediately performs a quantitative evaluation to abort a decision found to be highly likely to cause a break in the temporal reasoning chain. Then it uses the failed attempt as new feedback to explore the other more correct reasoning chains. This iterative process operates without any fine-tuning while enhancing interpretability. Empirical validation on the CronQuestions and MultiTQ benchmarks demonstrates that the overall semantic potential energy monotonically decreases as reasoning converges toward the correct answer.

Our main contributions are summarized as follows:

- We innovatively introduce the physical principle of minimum potential energy to address the problem of the broken temporal reasoning chain in TKGQA tasks.
- We propose the **Regret-Now** framework, which leverages a quantifiable semantic potential energy to perform real-time detection of broken reasoning chains and correct the LLM's erroneous reasoning decisions.
- We demonstrate that Regret-Now significantly outperforms strong baselines on the CronQuestions and MultiTQ benchmarks. We validate our core physics-inspired hypothesis by confirming that a decrease in potential energy leads to the convergence of reasoning.

2 Related Work

TKGQA. Early TKGQA methods primarily relied on rule decomposition (Jia et al., 2018) or knowledge graph embedding (Saxena et al., 2021; Chen et al., 2023b; Leblay and Chekol, 2018; Dasgupta et al., 2018). Although subsequent works improved performance by incorporating dedicated temporal modules (Abbasiantaeb et al., 2024; Mavromatis et al., 2022; Shang et al., 2022; Li et al., 2023a; Liu et al., 2023; Sharma et al., 2023; Jia et al., 2021), the common drawback of these traditional methods is their static reasoning paths. They lack the ability for real-time monitoring and dynamic correction during the reasoning process, making it difficult to salvage the entire reasoning chain once an intermediate step is erroneous.

LLM-based Temporal Reasoning. Recent studies have begun to leverage the powerful reasoning capabilities of LLMs (Sun et al., 2024), typically within agent-based frameworks combined with

retrieval tools (Yao et al., 2023b; Zhao et al., 2024; Wang et al., 2024; Abbasiantaeb et al., 2024). However, imprecise temporal understanding often leads to the retrieval of irrelevant noise. To mitigate this issue, methods like ARI (Chen et al., 2023a), GenTKGQA (Gao et al., 2024), and TempAgent (Hu et al., 2025) have attempted to enhance the model's temporal sensitivity. Nevertheless, these approaches are largely"result-oriente" and generally lack dynamic verification mechanisms within the reasoning process, which makes them highly susceptible to cascading errors.

Self-Correction in Multi-step Reasoning. To enhance reasoning performance, the research community has explored the self-correction capabilities of LLMs (Kim et al., 2023; Renze and Guven, 2024; Madaan et al., 2023; Paul et al., 2024). Building upon Chain-of-Thought (Wei et al., 2022), methods like Reflexion (Shinn et al., 2023) and Tree-of-Thoughts (Yao et al., 2023a) introduce mechanisms for reflection or search-backtracking. However, the fundamental limitation of these methods is that their correction decisions rely on the LLM's own, subjective evaluation (Huang et al., 2023). This endogenous evaluation standard can introduce bias and lacks an external, interpretable, and objective criterion to guarantee the correctness of the correction's direction.

3 Regret-Now

3.1 Overview

As illustrated in Figure 2, we propose a multi-step temporal reasoning framework, **Regret-Now**, inspired by the physics principle of *minimum potential energy*. The framework decomposes temporal QA into sequential decision steps:

Stage 1: Decision-making The LLM analyzes the current reasoning state and history to select the optimal next reasoning operation from multiple possible operation.

Stage 2: Regret The framework immediately performs a quantitative evaluation of the decision. If it is found that the current decision is highly likely to cause a break in the temporal reasoning chain, that decision is then aborted and negated.

Stage 3: Re-decision This stage uses the failed attempt as new feedback to guide the model to explore other, more correct reasoning chains.

Notably, Regret-Now operates without any fine-tuning or parameter updates. This design significantly reduces computational costs and facilitates plug-and-play integration with existing LLMs.

3.2 Stage 1: Decision-making

This stage draws inspiration from Chen et al. (2023a), primarily leveraging the inherent contextual understanding and generative capabilities of LLMs to make decisions at each step of temporal reasoning.

Candidate Operation Generation. Given a question q, we first obtain the subject entity e_h and construct its 1-hop subgraph G_{sub} in the TKG. From G_{sub}, we generate candidate operations C_0 by traversing a predefined template list for each neighboring entity–relation pair[1].

We then filter all candidate operations based on executability and semantic relevance to the question. The top-K operations form the refined candidate set C:

$$C = \{c \mid \textit{Executable(c)} \wedge c \in \textit{Top-K}(C_0, q)\}. \tag{1}$$

LLM Decision. We format the candidate operations C into a prompt and feed it to the LLM[2]:

$$decision_i = \textit{LLM}(q, h_{i-1}, C), \tag{2}$$

where at step i, the LLM selects the best operation $decision_i$ based on the input question q, the reasoning history up to step $i-1$, and the current candidate set C.

TKG Retrieval. Finally, the retriever executes the chosen decision operation $decision_i$ on the TKG to obtain the corresponding entities or timestamps as the output for this step.

3.3 Stage 2: Regret

3.3.1 Potential Energy

Inspired by the *principle of minimum potential energy*, we view temporal reasoning on TKGs as a process where each reasoning step should lead to progressively lower potential energy. The final answer corresponds to the system's energy minimum, i.e., its most stable state.

We model entities as particles and relations as physical forces (e.g., gravity or springs), thereby transforming the existing TKG into a stable physical system. At each reasoning step, we compute the

[1]The optimized full list of candidate operation templates is in Appendix B.1.

[2]See Appendix F for prompt details.

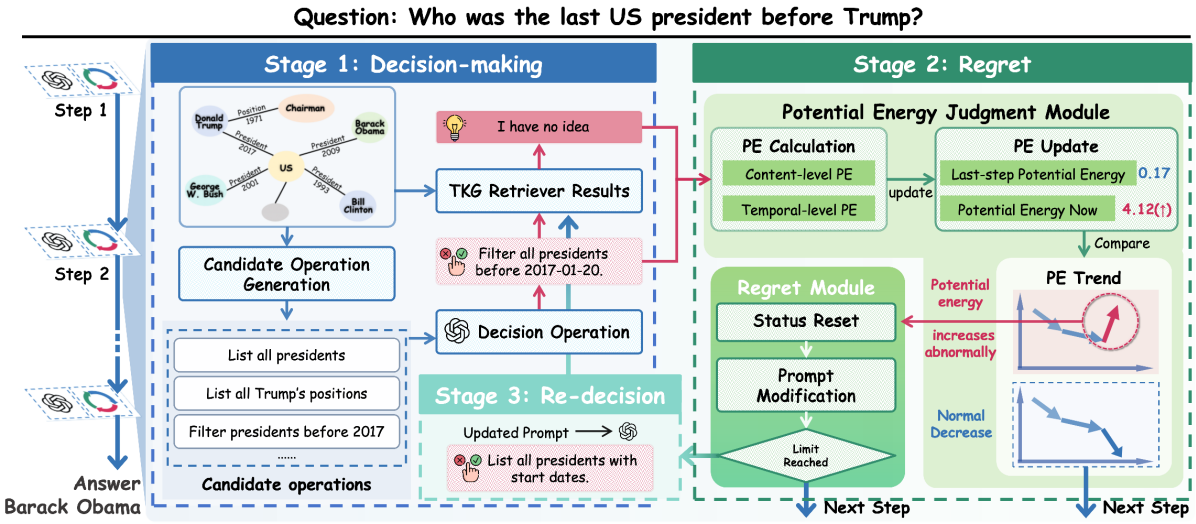


Figure 2: Overview of the Regret-Now Framework. The red arrow indicates that the potential energy has increased abnormally after the potential energy judgment module, triggering the regret module. The green arrow indicates that the potential energy has decreased as expected, entering the next reasoning step.

potential energy U based on Newton's law of universal gravitation and the associated relationship between force and potential, as shown below:

$$F = G\frac{m_1 m_2}{r^2} = -\nabla U. \tag{3}$$

Based on Equation (3), the corresponding potential energy U is:

$$U = -G\frac{m_1 m_2}{r}. \tag{4}$$

Here, G is the gravitational constant, m_1 and m_2 are particle masses, and r is the semantic distance between entities. F is the resulting semantic force, U is the potential energy representing reasoning stability. Since each decision may retrieve multiple result entities, we introduce a *virtual particle* representing their average embedding position. Thus, m_1 and m_2 represent the masses of the subject entity and the virtual entity, respectively. The term r denotes their distance in the embedding space.

$$\mathbf{u}_e^{(k)} = \frac{1}{|\mathcal{E}^{(k)}|}\sum_{e\in\mathcal{E}^{(k)}} \mathbf{u}_e, \tag{5}$$

where $\mathbf{u}_e^{(k)}$ is the virtual particle embedding used in subsequent energy computation, and $\mathcal{E}^{(k)}$ denotes the set of retrieved entities at step k.

We compute the semantic distance r using embedding representations obtained from the TComplEx model (Lacroix et al., 2020) for TKG and from CronKGQA (Saxena et al., 2021) for questions. Since semantically closer entities tend to have smaller distances in the embedding space, we define:

$$r(u_s, u_e, q, t) = \frac{1}{\mathrm{Re}\left(\langle \mathbf{u}_s, \mathbf{q}, \mathbf{u}_e^*, \mathbf{w}_t\rangle\right) + \epsilon}, \tag{6}$$

where u_s and u_e denote the embeddings of the subject entity and the current virtual entity, w_t is the temporal embedding from TComplEx, and q is the question embedding from CronKGQA. $\mathrm{Re}(\cdot)$ extracts the real part of the above multi-linear dot product of embeddings, and ϵ is a positive constant ensuring that the distance r remains strictly positive. A theoretical justification for using $\mathrm{Re}(\cdot)$ is provided in Appendix C.

3.3.2 Potential Energy Judgment Module

We decompose potential energy into two dimensions, content-level (U_{ent}) and temporal-level (U_{time}), to disentangle and precisely capture deviations within the intended reasoning dimension, thereby improving interpretability and reasoning evaluation capability:

$$U_{\text{ent}} = -Gm_1m_2\left(\mathrm{Re}\left(\langle \mathbf{u}_s, \mathbf{q}_{\text{ent}}, \mathbf{u}_e^*, \mathbf{w}_t\rangle\right) + \epsilon\right), \tag{7}$$

$$U_{\text{time}} = -Gm_1m_2\left(\mathrm{Re}\left(\langle \mathbf{u}_s, \mathbf{q}_{\text{time}}, \mathbf{u}_e^*, \mathbf{w}_t\rangle\right) + \epsilon\right), \tag{8}$$

where $\mathbf{q}_{\text{ent}}$ and $\mathbf{q}_{\text{time}}$ are the question-guided query embeddings for entity reasoning and temporal reasoning, respectively; $\mathbf{w}_t$ is the temporal embedding.

Since entities in TKGs are treated symmetrically and G is a constant, one could in principle as-

sign different values of m_1 and m_2 based on structural priors such as PageRank, in-degree, thereby reflecting different "masses" of entities. However, to focus on the core mechanism of Regret-Now—detecting abnormal potential energy fluctuations and triggering rollback, we adopt a simplified normalization by setting $m_1 = m_2 = \frac{1}{\sqrt{G}}$. Under this setting, U_{ent} and U_{time} depend only on embedding interactions, streamlining subsequent analysis while keeping the physical analogy intact.

When the LLM decides on a timestamp-related operation (e.g., *"get time"*), we select U_{time} as the potential energy ; for entity-focused operations (e.g., *"get head"*), we select U_{ent}. This dimension-specific formulation allows Regret-Now to monitor energy fluctuations separately, ensuring that rollback is triggered only when deviations arise in the relevant reasoning dimension.

$$U^{(k)} = \begin{cases} U^{(k)}_{\text{time}}, & o^{(k)} \in \mathcal{O}_{\text{time}}, \\ U^{(k)}_{\text{ent}}, & o^{(k)} \in \mathcal{O}_{\text{ent}}, \end{cases} \tag{9}$$

where $o^{(k)}$ is the selected operation at step k, and $\mathcal{O}_{\text{time}}$ / $\mathcal{O}_{\text{ent}}$ denote the timestamp-oriented and entity-oriented operation sets.

We monitor the energy increment at each step, where k denotes the current step:

$$\Delta U^{(k)} = U^{(k)} - U^{(k-1)} > \tau. \tag{10}$$

When the potential energy increases abnormally, the **Regret Module (RM)** is triggered, where τ denotes the corresponding threshold.

3.3.3 Regret Module

Once an abnormal energy ascent is detected, the Regret Module is triggered to perform a rollback operation, resetting the reasoning process to the step before the erroneous decision.

Concretely, Regret Module first clears the intermediate reasoning trace of the current step and marks it as a "regretting state", indicating that the prior decision has deviated from the natural descent trajectory. The failed operation then explicitly injected into the next prompt, allowing the model to reconsider the same decision context while being aware of its previous failure.

To avoid infinite loops from repeated hallucinations, we cap the maximum number of regrets per step.

3.4 Stage 3: Re-decision

After the Regret Module is triggered, the framework enters the Re-decision stage. Here, the LLM is re-invoked with an updated prompt that includes: (1) the original question, (2) the revised history with the previous decision rolled back, and (3) an explicit indication of the failed operation and the regret signal. The updated prompt is detailed in Appendix F. This contextual modification encourages the model to reconsider its reasoning chain and select a more plausible operation.

4 Experiments

This section presents a comprehensive evaluation of the proposed **Regret-Now** framework for TKGQA. Our evaluation is structured around the following research questions (RQs):

- **RQ1:** How does **Regret-Now** perform compared to existing baselines?
- **RQ2:** Does potential energy truly decrease during the process of reasoning toward the correct answer?
- **RQ3:** When and where does regret tend to occur?
- **RQ4:** Can **Regret-Now** remain efficient while benefiting from the Regret Stage?

4.1 Experimental Setup

4.1.1 Dataset

We evaluate on two large-scale temporal QA datasets: **CronQuestions** (Saxena et al., 2021) with 410K year-level questions from Wikidata, and **MultiTQ** (Chen et al., 2023b) with 500K multi-hop questions requiring complex temporal reasoning [3].

4.1.2 Evaluation Metrics

Following prior work (Chen et al., 2023a; Li et al., 2023b; Hu et al., 2025), we report Hit@1, where the model outputs a single answer and is evaluated based on exact match with the gold label.

4.1.3 Baselines

We benchmark Regret-Now against baselines from three categories: foundational pre-trained LMs, specialized TKGQA models, and LLM-based models.

- **Pre-trained LMs. BERT** (Devlin et al., 2019) and **ALBERT** (Lan et al., 2019) serve as fundamental baselines, representing the capabilities of non-specialized models trained on general text corpora.

[3] Dataset statistics and examples are provided in Appendix A.1.

Model Type	Models	CronQuestions					MultiTQ				
		Overall	Question Type		Answer Type		Overall	Question Type		Answer Type	
			Simple	Complex	Entity	Time		Single	Multiple	Entity	Time
Pre-trained LMs	BERT	0.243	0.249	0.239	0.277	0.179	0.083	0.092	0.061	0.101	0.040
	ALBERT	0.248	0.255	0.235	0.279	0.177	0.108	0.116	0.086	0.139	0.032
TKGQA Models	EmbedKGQA	0.288	0.290	0.286	0.411	0.057	0.206	0.235	0.134	0.290	0.001
	CronKGQA	0.647	0.987	0.392	0.699	0.549	0.279	0.134	0.134	0.328	0.156
	MultiQA	-	-	-	-	-	0.293	0.347	0.159	0.349	0.157
	TSQA	0.831	**0.987**	0.713	0.829	0.836	-	-	-	-	-
LLM-based Models	Naive RAG	0.633	0.726	0.280	0.610	0.684	0.378	0.469	0.155	0.242	0.672
	ARI	0.707	0.860	0.570	0.660	0.800	0.380	0.680	**0.210**	0.394	0.344
	TKG-RAG	0.723	0.745	0.701	0.789	0.617	0.244	0.292	0.117	0.281	0.131
Self-Correction	ToT	0.318	0.295	0.317	0.245	0.821	0.129	0.161	0.039	0.157	0.052
	ReAct-RAG	0.780	0.867	0.693	0.779	0.782	0.228	0.262	0.125	0.198	0.292
	Reflexion	0.815	0.907	0.728	0.813	0.818	0.271	0.312	0.148	0.235	0.345
Regret-Now	Regret-Now (w/o RS)	0.848	0.966	0.769	0.778	0.968	0.504	0.617	0.148	0.493	**0.534**
	Regret-Now(w/o PE-Sel)	0.850	0.959	0.792	0.793	0.968	0.455	0.594	0.146	0.473	0.341
	Regret-Now (ours)	**0.876**	0.979	**0.805**	**0.815**	**0.971**	**0.543**	**0.693**	0.192	**0.577**	0.380

Table 1: Performance comparison across the CronQuestions and MultiTQ datasets. **Regret-Now (ours)** uses GPT-4o as the LLM backbone. **Regret-Now (w/o PE-Sel)** removes multi-level PE selection; **Regret-Now (w/o RS)** removes the Regret Stage. **Bold** denotes the best, and underlined the second-best result in each column.

- **TKGQA Models.** We include models explicitly engineered for TKGQA. We include the static **EmbedKGQA** (Saxena et al., 2020) and a series of progressively advanced temporal models: **CronKGQA** (Saxena et al., 2021) (time constraints), **MultiQA** (Chen et al., 2023b) (multi-granularity timestamps), and **TSQA** (Shang et al., 2022) (contrastive learning).

- **LLM-based Models.** We benchmark a spectrum of LLM-based reasoners, categorizing them by the presence of a self-correction mechanism. (1) LLM-based Models without Self-Correction, such as **ARI** (Chen et al., 2023a), which leverages a modular instruction design, and retrieval-augmented methods like **Naive RAG** and **TKG-RAG** that ground the model's reasoning in structured temporal knowledge. (2) Models with Self-Correction, including **ToT** (Yao et al., 2023a), which systematically explores a tree of thought processes; **ReAct-RAG** (Yao et al., 2023b), which integrates a feedback loop between reasoning, acting, and observation; and **Reflexion** (Shinn et al., 2023), which enables the model to self-critique and revise its reasoning.

4.1.4 Hyper-parameter Setting

Our framework uses two key hyper-parameters. A potential energy threshold of 5 triggers the Regret Module when energy rises sharply between steps. We also allow up to 2 regret attempts per step to improve stability against repeated local errors.

4.2 Main Results

4.2.1 Overall Performance (RQ1)

Table 1 shows that Regret-Now achieves competitive or superior results against baselines on CronQuestions and MultiTQ. To better understand these results, we further conduct a fine-grained analysis by categorizing questions according to their type and answer type [4]. Regret-Now achieves its significant gains on entity-type answers. Within this category, the improvements are particularly pronounced on questions of the "After First" and "Before Last" types, which require complex multi-step temporal reasoning. We attribute this strong performance to the proposed potential energy judgment mechanism, which enables the model to explicitly detect abnormal reasoning transitions and prune erroneous reasoning chains at an early stage. In contrast to existing LLM-based self-correction methods that rely on subjective model self-evaluation, our mechanism grounds correction decisions in an explicit and interpretable criterion, leading to more reliable identification and correction of erroneous reasoning actions. However, Regret-Now struggles with time-type questions in MultiTQ due to a mismatch between our day-level embeddings and the dataset's multi-granular nature (years, months, and days). This discrepancy prevents the accurate assessment of particle positions for non-day granularities, causing performance degradation on the specific answer type.

[4] See Appendix A.1 for examples.

Base LLM	MultiTQ				
	Overall	Question Type		Answer Type	
		Single	Multiple	Entity	Time
Llama-3	0.451	0.566	0.120	0.472	0.364
Deepseek-v3	0.437	0.564	0.126	0.453	0.366
Qwen3	0.340	0.429	0.075	0.361	0.268
GPT-4o	**0.543**	**0.693**	**0.192**	**0.577**	**0.380**

Table 2: Performance of Regret-Now with Different Base LLMs on the MultiTQ Dataset.

4.2.2 Ablation Study

Regret Module. We conducted an ablation study on the Regret Stage (RS) by removing it from our model (Regret-Now w/o RS). As Table 1 shows, this ablation caused significant performance drops of 2.8% on CronQuestions and 3.9% on MultiTQ. The degradation was widespread, with drops on some question types exceeding 8% and a notable 7.2% decline on MultiTQ's most complex queries. Given that MultiTQ requires more sophisticated temporal reasoning, these results strongly validate that the Regret Stage is vital for the model's ability to tackle complex temporal reasoning tasks.

Selection of Potential Energy. We ablated our adaptive potential energy (PE) selection strategy by replacing it with a static, simple average of the content and temporal PE components (Regret-Now w/o PE-Sel). This change forces a fixed energy representation. As per Table 1, this ablation caused performance to degrade by 2.6% on CronQuestions and 8.8% on MultiTQ. The decline is more pronounced in MultiTQ demonstrates that a static average is insufficient. These results confirm the necessity of separating potential energy into content and temporal dimensions and adapting their usage according to specific reasoning needs.

Effect of Different Base LLMs. We tested Regret-Now with four different base LLMs on MultiTQ. The results in Table 2 show that the GPT-4o variant delivered the highest performance. This finding highlights a key synergistic relationship: Regret-Now effectively leverages the advanced capabilities of the base model. As the base LLM's capacity for temporal understanding and complex reasoning improves, the performance gains afforded by our method are amplified.

4.2.3 Analysis of Potential Energy During Inference (RQ2)

To validate the principle of minimum potential energy, we conducted an empirical analysis of its dynamics during inference. The results, shown in Figure 3, reveal a consistent downward trend in the average potential energy across all correct reasoning trajectories, indicating that the inference process naturally gravitates towards lower-energy states. Transient energy increases, predominantly in the early stages, are attributable to noisy retrievals that introduce initial directional noise. Nevertheless, as the reasoning process unfolds, these deviations are systematically corrected. The progressive pruning of candidate entities leads to a sustained energy reduction. This self-correcting convergence is further substantiated by a diminishing step-level regret trigger rate, reinforcing the theoretical validity of the potential energy model.

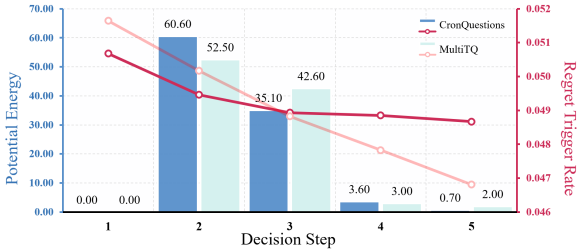


Figure 3: Potential Energy Trends (line) and Regret Trigger Rates (bar) over Decision Steps on CronQuestions and MultiTQ.

Rate	CronQuestions				
	Overall	Question Type		Answer Type	
		Simple	Complex	Entity	Time
RM Trigger Rate	0.315	0.147	0.422	0.303	0.304
Post-Correction Accuracy	0.709	0.929	0.651	0.536	0.981

Table 3: Regret Module Trigger Rate and Post-Correction Accuracy on the CronQuestions Dataset.

4.2.4 When to Regret: Regret Module Trigger Analysis (RQ3)

As shown in Table 3, the RM is triggered in 31.5% of cases with a post-trigger accuracy of 70.9%. Its higher trigger rate on complex questions (42.2%) confirms that complex reasoning is more error-prone.

To pinpoint the failure modes of LLMs, we analyze the distribution of RM-triggered steps. Our analysis, illustrated in Figure 3, reveals that errors predominantly occur during the 2nd and 3rd steps of multi-step reasoning across both CronQuestions and MultiTQ. Notably, these error peaks in trigger rates align with inflection points on the potential energy curves. This alignment indicates that the RM effectively identifies steps with high energetic instability, thereby validating potential energy as a fine-grained signal for detecting uncertainty in the reasoning process.

Model	Accuracy	Avg. Steps
ToT	0.318	5.73
ReAct-RAG	0.780	2.14
Reflexion	0.815	2.64
Regret-Now (w/o RS)	0.848	2.77
Regret-Now (ours)	0.876	2.92

Table 4: Efficiency Comparison of Regret-Now with and without Regret Stage.

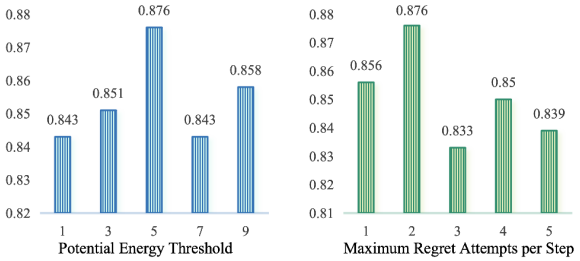


Figure 4: Impact of Key Hyper-parameters on the Performance of Regret-Now on CronQuestions: (a) Effect of Potential Energy Threshold on Performance; (b) Effect of Maximum Regret Attempts per Step.

4.2.5 Efficiency Evaluation of Regret-Now (RQ4)

We evaluate **Regret-Now** on the CronQuestions dataset against several representative self-correction methods, including ToT, ReAct-RAG, and Reflexion, as well as an ablated variant, Regret-Now (w/o RS). We measure both final accuracy and the average number of reasoning steps, which serves as a proxy for computational cost. As shown in Table 4, Regret-Now achieves the highest accuracy (0.876) among all methods, while requiring only a modest increase in reasoning steps (+0.15 over Regret-Now w/o RS, +0.28 over Reflexion).

4.2.6 Hyper-parameter Selection

Since early noise may cause temporary deviations from a strictly decreasing potential energy. We therefore vary the threshold and find that a value of 5 achieves the best performance on CronQuestions (Figure 4 (a)). We further analyze the maximum number of regret attempts per step. Allowing limited retries improves stability, but excessive ones can pollute the reasoning history. As shown in Figure 4 (b), setting the limit to 2 provides a balanced trade-off between correction and stability.

4.3 Case Study and Error Analysis

4.3.1 Case Study

Figure 5 illustrates how **Regret-Now** identifies and recovers from a reasoning failure. In contrast to the correct reasoning process, the model on the right prematurely imposed a temporal constraint, which caused the second-step retrieval to return

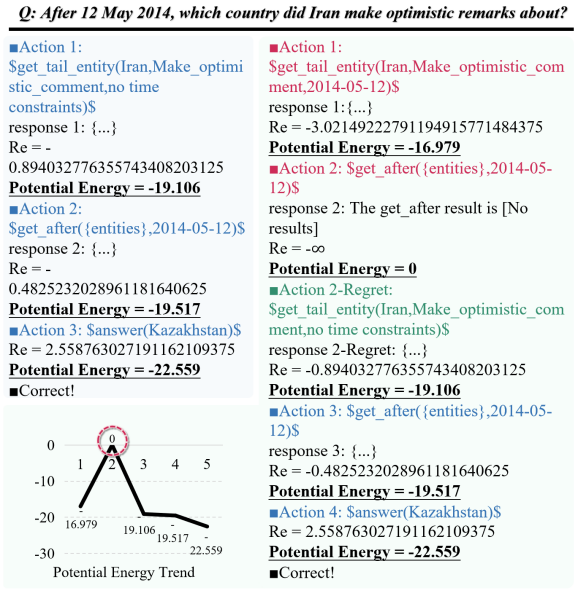


Figure 5: Case Study 1. Correct (Left) and Regret (Right) reasoning trajectory leading to the answer **Kazakhstan**, with steadily decreasing potential energy.

an empty set. This logical error is reflected by the potential energy sharply rising to 0. After the system detected the anomaly, it triggered the regret stage, rolling back to adopt the correct chain, and ultimately succeeded. We also present examples of cases where potential energy remains unchanged in the Appendix E.

4.3.2 Error Analysis

Two major failure types emerge: (1) **Argument role confusion**, where LLMs invert subject–object roles (e.g., "Iran praised X" vs. "X praised Iran"); and (2) **Suboptimal decisions**, where models choose lower-probability operations despite the highest-prbability operation is correct, reflecting hallucination or prompt sensitivity. These findings indicate that LLM reasoning is not fully confidence-aligned, suggesting the need for control mechanisms during generation as well as post-hoc regret detection. Detailed Examples and token-level analyses are provided in Appendix E.

5 Conclusion

We propose **Regret-Now**, a physics-inspired framework for multi-step temporal reasoning in LLMs. By introducing the *principle of minimum potential energy*, each reasoning step is evaluated via a potential energy metric. A Regret Stage monitors for abnormal energy increases and triggers real-time rollbacks to revise faulty decisions. Experiments on CronQuestions and MultiTQ demonstrate that Regret-Now consistently improves performance.

6 Limitations

Building on the promising results discussed above, several directions remain open for future exploration. (1) **Exploring potential energy representation in more dimensions: Regret-Now** currently relies on predefined embedding methods for entities and questions to estimate potential energy. Future work may explore more expressive and flexible dimensions—incorporating semantic, temporal, structural, and entity-specific attributes—to enrich the modeling space of potential energy. (2) **Greater adaptability:** The current modeling of potential energy is relatively simplistic, employing a uniform temporal embedding scheme that may fall short in handling questions involving varying levels of temporal granularity. Enhancing the robustness and accuracy of energy estimation calls for more sophisticated temporal representations.

Acknowledgements

This project has received funding from the National Natural Science Foundation of China (Nos. 62306156, 72571150, and 62106091), the Shandong Provincial Natural Science Foundation (No. ZR2025MS1078), and the Tianjin Municipal Science and Technology Bureau (No. 25JCZDSN00020).

References

Zahra Abbasiantaeb, Yifei Yuan, Evangelos Kanoulas, and Mohammad Aliannejadi. 2024. Let the llms talk: Simulating human-to-human conversational qa via zero-shot llm-to-llm interactions. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 8–17.

Ziyang Chen, Dongfang Li, Xiang Zhao, Baotian Hu, and Min Zhang. 2023a. Temporal knowledge question answering via abstract reasoning induction. *arXiv preprint arXiv:2311.09149*.

Ziyang Chen, Jinzhi Liao, and Xiang Zhao. 2023b. Multi-granularity temporal question answering over knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 11378–11392.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2001–2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 4171–4186.

Yifu Gao, Linbo Qiao, Zhigang Kan, Zhihua Wen, Yongquan He, and Dongsheng Li. 2024. Two-stage generative question answering on temporal knowledge graph using large language models. *arXiv preprint arXiv:2402.16568*.

Qianyi Hu, Xinhui Tu, Guo Cong, and Shunping Zhang. 2025. Time-aware react agent for temporal knowledge graph question answering. In *Proceedings of the Findings of Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, pages 6013–6024.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics*, page 1049–1065.

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

Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018. Tequila: Temporal question answering over knowledge bases. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1807–1810.

Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Complex temporal question answering on knowledge graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 792–802.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 36, pages 39648–39677.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. *arXiv preprint arXiv:2004.04926*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion proceedings of the the web conference 2018*, pages 1771–1776.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023a. Few-shot in-context learning for knowledge base question answering. *arXiv preprint arXiv:2305.01750*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023b. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*.

Ruotong Liao, Xu Jia, Yunpu Ma, and Volker Tresp. 2023. Gentkg: Generative forecasting on temporal knowledge graph. *arXiv preprint arXiv:2310.07793*.

Yonghao Liu, Di Liang, Mengyu Li, Fausto Giunchiglia, Ximing Li, Sirui Wang, Wei Wu, Lan Huang, Xiaoyue Feng, and Renchu Guan. 2023. Local and global: Temporal question answering via information fusion. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, pages 5141–5149.

Ruilin Luo, Tianle Gu, Haoling Li, Junzhe Li, Zicheng Lin, Jiayi Li, and Yujiu Yang. 2024. Chain of history: Learning and forecasting with llms for temporal knowledge graph completion. *arXiv preprint arXiv:2401.06072*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback, 2023. *URL https://arxiv.org/abs/2303.17651*.

Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N Ioannidis, Adesoji Adeshina, Phillip R Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. 2022. Tempoqr: temporal question reasoning over knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 5825–5833.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. Refiner: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1100–1126.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. 2021. Question answering over temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, page 6663–6676.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

Chao Shang, Guangtao Wang, Peng Qi, and Jing Huang. 2022. Improving time sensitivity for question answering over temporal knowledge graphs. *arXiv preprint arXiv:2203.00255*.

Aditya Sharma, Apoorv Saxena, Chitrank Gupta, Mehran Kazemi, Partha Talukdar, and Soumen Chakrabarti. 2023. Twirgcn: Temporally weighted graph convolution for question answering over temporal knowledge graphs. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2049–2060.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2024. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 9828–9862.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Zeqi Zhou, Fang Wu, Shayan Talaei, Haokai Zhao, Cheng Meixin, Tinson Xu, Amin Saberi, and Yejin Choi. 2025. When to trust context: Self-reflective

debates for context reliability. *arXiv preprint arXiv:2506.06020*.

A Dataset Statistics and Baseline Descriptions

A.1 Dataset Statistics

We conduct experiments on two temporal question answering datasets: **CronQuestions** (Saxena et al., 2021) and **MultiTQ** (Chen et al., 2023b). Following common practice and the amount of data used in (Hu et al., 2025), and given that running full-scale evaluations across all datasets would require considerable LLM computational resources due to multi-step temporal reasoning, we address our limited computational budget by randomly sampling a subset of 300 questions from each dataset while maintaining the original distribution of question types. All experiments, aside from the main results in Table 1, are conducted on this subset.

A.1.1 CronQuestions

This dataset focuses on temporal knowledge graph question answering, where each question is annotated with both entity and time constraints. As shown in Table 5, questions are categorized into five types: *Simple Entity*, *Simple Time*, *Before/After*, *First/Last*, and *Time Join*. The dataset further groups all questions into two broad categories based on their complexity: *Simple questions*, including *Simple Entity* and *Simple Time*, and *Complex questions*, comprising *Before/After*, *First/Last*, and *Time Join*. Table 7 shows representative examples for each reasoning type. In total, the dataset contains 410K questions, split into 350K/30K/30K for train/dev/test, respectively.

Category	Train	Dev	Test
Simple Entity	90,651	7,745	7,812
Simple Time	61,471	5,197	5,046
Before/After	23,869	1,982	2,151
First/Last	118,556	11,198	11,159
Time Join	55,453	3,878	3,832
Entity Answer	225,672	19,362	19,524
Time Answer	124,328	10,638	10,476
Total	**350,000**	**30,000**	**30,000**

Table 5: Detailed statistics of CronQuestions by question and answer type.

A.1.2 MultiTQ

This dataset extends temporal QA with large-scale, diverse, and fine-grained temporal information. As shown in Table 6, questions are categorized into six types: *Equal*, *Before/After*, *First/Last*, *Equal Multi*, *After First*, and *Before Last*. Similar to CronQuestions, all questions are grouped into two categories: *Simple questions*, including *Equal*, *Before/After*, and *First/Last*, and *Complex questions*, including *Equal Multi*, *After First*, and *Before Last*, which require multi-step reasoning. Table 8 presents representative examples categorized by question type, time granularity (e.g., year, month, day), and answer type (entity/time). In total, MultiTQ contains over 500K questions, with 386K/58K/55K for train/dev/test, respectively.

Category	Question Type	Train	Dev	Test
Single	Equal	135,890	18,983	17,311
	Before/After	75,340	11,655	11,073
	First/Last	72,252	11,097	10,480
Multiple	Equal Multi	16,893	3,213	3,207
	After First	43,305	6,499	6,266
	Before Last	43,107	6,532	6,247
Total		**386,787**	**57,979**	**54,584**

Table 6: Statistics of question categories in MultiTQ.

A.2 Baseline Descriptions

We compare **Regret-Now** against a comprehensive set of baselines, covering three main categories: pre-trained LMs, embedding-based TKGQA methods, and LLM-based reasoning frameworks.

Pre-trained LMs We evaluate two representative language models, **BERT** (Devlin et al., 2019) and **ALBERT** (Lan et al., 2019), by encoding the input question and computing a dot-product score between the resulting embedding and all candidate entity/time embeddings.

Embedding-based TKGQA These methods operate on temporal knowledge graphs by learning structured embeddings:

- **EmbedKGQA** (Saxena et al., 2020) answers questions by aligning them with entity-relation pairs in static knowledge graphs. It does not explicitly handle temporal constraints.
- **CronKGQA** (Saxena et al., 2021) extends the embedding-based approach by introducing time-aware representations.
- **MultiQA** (Chen et al., 2023b) extends temporal embedding methods to support multi-granularity timestamps, and introduces a transformer-based module for aggregating

Reasoning	Example Question
Simple time	*When did Obama hold the position of President of USA*
Simple entity	*Which award did Brad Pitt receive in 2001*
Before/After	*Who was the President of USA before Obama*
First/Last	*When did Messi play their first game*
Time join	*Who held the position of President of USA during WWII*

Table 7: Representative reasoning types and example questions from CronQuestions.

Property	Sample Question
By question type	
Equal	Which country provided humanitarian aid to Sudan in 2007?
Before/After	Who commended the Military of Mali before the Armed Rebel of Mali did?
First/Last	When did the Militant of Taliban first commend the Government of Pakistan?
Equal Multi	In 2012, who last did Barack Obama appeal for?
Before Last	Who was threatened by Benjamin Netanyahu last before Middle East?
After First	Who first wanted to negotiate with Evo Morales after the Citizen of Brazil did?
By time granularity	
Year	Who first made Abu Sayyaf suffer from conventional military forces in 2015?
Month	In Dec, 2008, who would wish to negotiate with the Senate of Romania?
Day	In Jul 21st, 2011, who criticized the Media of Ecuador?
By answer type	
Entity	Which country visited Japan in 2013?
Time	When did China express intent to meet with the Government of Pakistan?

Table 8: Representative examples from MultiTQ.

temporal information across varying time scales.

- **TSQA** (Shang et al., 2022) introduces a time estimation module and contrastive time-order learning to capture implicit temporal references and enhance sensitivity to temporal relations in questions.

LLM-based TKGQA These methods leverage large language models as reasoning agents over temporal KGs:

- **ToT** (Yao et al., 2023a) extends the chain-of-thought paradigm by organizing reasoning as a search process over a tree structure. At each step, the model generates multiple candidate thoughts (*propose*), evaluates their promise (*evaluate*), and selects the most promising branches to expand (*select*). This iterative exploration enables the model to maintain and refine multiple reasoning paths in parallel, effectively mitigating early decision errors and improving stability on multi-step reasoning tasks. In our implementation, ToT is instantiated using the **GPT-4o** backbone.
- **ARI** (Chen et al., 2023a) enhances the agent by introducing abstract, modular reasoning instructions. It improved performance on complex multi-step questions. We implement ARI with the **GPT-4o** backbone.
- **TKG-RAG** retrieves context by using the subject entity as a query anchor to collect temporally related facts from the TKG. These facts are concatenated and provided to the LLM (**GPT-4o**) for direct answer generation. When the retrieved content exceeds the input limit, a subset is randomly sampled within the token budget.
- **ReAct-RAG** (Yao et al., 2023b) extends ReAct by integrating KG feedback into the LLM's reasoning–action loop, where the LLM backbone is **GPT-4o**.
- **Reflexion** (Shinn et al., 2023), a self-correction framework that enables the model to iteratively critique and revise its own reasoning trajectories based on feedback from previous attempts. In our implementation, the reflection mechanism is realized within the ReAct reasoning framework, with **GPT-4o**

serving as the LLM backbone.

B Model Configuration Details

B.1 Full list of operation templates

Table 9 lists all the operation templates used during step-wise reasoning. Each template corresponds to a primitive reasoning operation executable by the model, including entity retrieval, temporal filtering, comparative operations, and final answer generation. These operations are adapted and extended from the operation design proposed by ARI, with improvements for better alignment with potential energy evaluation and rollback-based correction. They serve as the atomic units of our reasoning framework and are invoked sequentially during multi-step inference.

B.2 Details of Different LLM Bases Selected

To evaluate the generality and compatibility of **Regret-Now**, we instantiate our framework using four different large language models (LLMs) as the base reasoner:

- **Llama-3 (Sonar-Small-32K-Chat)**: A chat-optimized variant of Meta's LLaMA-3 family, provided by the Sonar suite. It features a compact architecture with 8B parameters and a 32K context window, and is optimized for instruction-following and long-context reasoning.
- **DeepSeek-V3**: A 16B multilingual model developed by DeepSeek AI. It is trained with emphasis on multi-turn alignment and factual accuracy, making it suitable for structured reasoning over temporal questions.
- **Qwen3-14B (Chat)**: A high-capacity open-source model released by Alibaba Cloud, pre-trained on diverse corpora with dialogue and tool-use capabilities. The chat version used in our experiments demonstrates strong performance in knowledge-centric tasks.
- **GPT-4o**: OpenAI's latest flagship model, designed for high efficiency and multi-modal interaction. Despite being closed-source, GPT-4o provides the most robust performance among all models tested, and serves as an upper-bound reference for our framework.

All models are accessed via their official APIs or open checkpoints, and integrated into our framework without additional fine-tuning.

B.3 LLM Configuration

All methods in our framework are instantiated using the GPT-4o model. We use a temperature of 0.7 with nucleus sampling ($top\text{-}p = 1.0$) for generation.

C Theoretical Basis for Using $\mathrm{Re}(\cdot)$ as Distance r

To interpret the real part of the complex product as a similarity measure, we start from the standard cosine similarity definition in vector spaces. Consider two complex vectors $z_1 = a_1 + ib_1$ and $z_2 = a_2 + ib_2$, with real parts a_i and imaginary parts b_i.

The cosine similarity between them is given by:

$$\cos\theta = \frac{\langle z_1, z_2 \rangle}{\|z_1\|\|z_2\|},$$

where $\langle z_1, z_2 \rangle$ denotes the inner product.

For the real inner product:

$$\langle z_1, z_2 \rangle = a_1 a_2 + b_1 b_2.$$

Thus, we have:

$$\cos\theta = \frac{a_1 a_2 + b_1 b_2}{\|z_1\|\|z_2\|}.$$

Now consider the complex product:

$$z_1 \overline{z_2} = (a_1 + ib_1)(a_2 - ib_2) = a_1 a_2 + b_1 b_2 + i(a_2 b_1 - a_1 b_2),$$

and its real part is:

$$\mathrm{Re}(z_1 \overline{z_2}) = a_1 a_2 + b_1 b_2.$$

Recall that the cosine of the angle θ between two real-valued vectors is defined as:

$$\cos\theta = \frac{a_1 a_2 + b_1 b_2}{\|z_1\|\|z_2\|},$$

where $\|z_1\| = \sqrt{a_1^2 + b_1^2}$ and $\|z_2\| = \sqrt{a_2^2 + b_2^2}$ denote the magnitudes of the complex numbers.

Thus, we conclude:

$$\mathrm{Re}(\langle z_1, z_2 \rangle) = \mathrm{Re}(z_1 \overline{z_2}) = \|z_1\|\|z_2\| \cos\theta.$$

This shows that $\mathrm{Re}(\langle z_1, z_2 \rangle)$ is positively correlated with $\cos\theta$. As $\mathrm{Re}(\langle z_1, z_2 \rangle)$ increases, so does the cosine similarity, which implies that:

$\mathrm{Re}(\langle z_1, z_2 \rangle)$ can serve as a proxy for similarity or alignment between z_1 and z_2.

Operation Template	Comments
getTailEntity(head, rel, time)	Identify the tail/object entity based on the head/subject entity and relation
getHeadEntity(tail, rel, time)	Identify the head/subject entity based on the tail/object entity and relation
getTime(head, rel, tail)	Retrieve the time of a specific event based on the head entity, relation and tail entity
getBetween(entities, time1, time2)	Identify entities/events that occurred between two specific times
getOverlap(entities, time1, time2)	Identify entities/events that occurred overlap two specific times
getBefore(entities, time)	Identify entities/events that occurred before a given time
getAfter(entities, time)	Identify entities/events that occurred after a given time
getFirst(entities, time)	Pinpoint entities with the earliest occurrence
getLast(entities, time)	Pinpoint entities with the latest occurrence
answer(entities/time)	To provide your answer, use the `answer` function

Table 9: List of primitive operation templates and their descriptions.

The distance r between two entities should decrease as their semantic similarity increases. Therefore, in our case, we adopt Equation 11 as a physically grounded quantity to reflect similarity, which serves as the theoretical foundation for potential energy computation in our framework.

$$r = \frac{1}{\mathrm{Re}\left(\langle e_1, e_2, q, t\rangle\right)} \tag{11}$$

$$r = \frac{1}{\mathrm{Re}\left(\langle e_1, e_2, q, t\rangle\right) + \epsilon} \tag{12}$$

Since the distance r is strictly positive, we introduce a small constant ϵ to ensure the denominator remains positive. The final formulation of r is given in Equation 12.

D Additional Experimental Results

D.1 Extended Results on Hyperparameter Selection

We further conduct hyperparameter selection experiments on the MultiTQ dataset.

Similar to CronQuestions, the reasoning process in MultiTQ involves gradual refinement toward the correct answer, which can be disrupted by early-stage decision noise. To mitigate this, we explore various threshold values for triggering the regret module. The results, summarized in Figure 6 (a), indicate that a threshold of 5 achieves the best balance between recall of errors and avoidance of unnecessary rollbacks. This result aligns with our earlier findings and suggests that the potential energy signal generalizes well across datasets.

We also investigate the effect of limiting the maximum number of regret attempts per reasoning step on MultiTQ. As shown in Figure 6 (b), allowing up to 2 attempts per step yields the highest accuracy, while both lower and higher settings lead to performance drops.

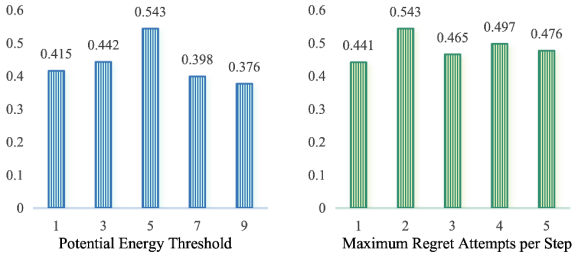


Figure 6: Impact of Key Hyper-parameters on the Performance of Regret-Now on MultiTQ: (a) Effect of Potential Energy Threshold on Performance; (b) Effect of Maximum Regret Attempts per Step.

D.2 Improvement or additional reasoning opportunities?

We collected all cases in which the regret module was triggered and led to a correct reasoning outcome, aiming to identify which reasoning steps are most error-prone for the LLM. As shown in Figure 7-(a), the results indicate that the LLM frequently makes mistakes during the 2nd and 3rd steps of multi-step reasoning.

To verify whether the performance improvement is solely due to providing the LLM with additional decision opportunities, we designed the following controlled experiments:

- **Regret-Now + random_disturb**: Randomly triggers the RM with a probability of 31.5%.
- **Regret-Now + step-level disturb**: Triggers the RM not based on potential energy evaluation, but rather according to the error-prone step distribution shown in Figure 7-(a). Given

Disturb Level	CronQuestions				
	Overall	Question Type		Answer Type	
		Simple	Complex	Entity	Time
Regret-Now + random_disturb	0.838	0.955	0.759	0.766	0.957
Regret-Now + step-level disturb	0.863	0.960	0.797	0.799	0.968

Table 10: Evaluating the Impact of Random vs. Targeted Regret Triggering on Reasoning Performance

the overall RM trigger rate of 31.5%, we simulate this by triggering the regret module at each step with a probability proportional to its error frequency. For example, the regret module is triggered at step 2 with a probability of $31.5\% \times 52.4\%$, and at step 3 with $31.5\% \times 42.6\%$, and so on.

The results, shown in Table 10, demonstrate that *Regret-Now + random_disturb* significantly degrades performance, as randomly triggered regret interrupts originally correct reasoning paths, introducing new errors instead. The *Regret-Now + step-level disturb* setting shows slight improvement over the baseline but still underperforms compared to Regret-Now across all question types. This is because the probability distribution implicitly captures the steps where the LLM is prone to errors, and introducing regret at these moments can improve correctness. Therefore, our experiments strongly confirm that Regret-Now does not simply provide "extra reasoning chances", but rather identifies and corrects flawed reasoning paths through dynamic potential energy analysis, leading to more effective reasoning performance.

D.3 Choose TNTComplEx to get embeddings

To further verify the performance of our framework with respect to the underlying temporal knowledge graph (TKG) embedding model, we additionally experiment with **TNTComplEx** (Lacroix et al., 2020).

TNTComplEx, which explicitly accounts for the heterogeneity of predicates in temporal knowledge graphs. While some relations (e.g., *daughterOf*) remain constant over time, others (e.g., *hasOccupation*) may evolve dynamically. To capture this distinction, TnTComplEx decomposes the fact tensor into two components: one temporal and one non-temporal.

Specifically, we replace the TComplEx encoder used in the main experiments with TNTComplEx, while keeping all other settings identical. The semantic distance r is thus computed based on embeddings generated by TNTComplEx for temporal entities and times, and by CronKGQA for question representations. This comparison helps assess whether more temporally expressive embeddings further enhance the Regret-Now framework's ability to capture potential energy variations across reasoning steps.

Model	CronQuestions				
	Overall	Question Type		Answer Type	
		Simple	Complex	Entity	Time
TComplEx	0.876	0.979	0.805	0.815	0.971
TNTComplEx	0.894	0.980	0.815	0.849	0.965

Table 11: Comparison of TComplEx and TNTComplEx embeddings in the Regret-Now framework.

The results demonstrate that employing TnTComplEx leads to consistent improvements over TComplEx across both datasets, which is consistent with the inherent modeling capacity of the two embedding methods. This suggests that more expressive temporal embeddings can better capture fine-grained temporal correlations, resulting in smoother potential energy transitions and more reliable regret detection. Importantly, these gains are achieved without any additional architectural modification or fine-tuning, further supporting the flexibility and generalizability of the Regret-Now framework.

D.4 Potential Energy Analysis on MultiTQ

We replicate the same analysis on the MultiTQ dataset to assess whether the energy descent phenomenon generalizes to more complex, multi-hop temporal questions.

Following the same protocol, we conduct a step-level breakdown of energy change statistics across all reasoning paths. As shown in Figure 7-(b) 46.3% of steps exhibit a decrease in potential energy, 3.8% remain unchanged, and 49.9% show an increase. Compared to the CronQuestions dataset, where energy descent was more dominant, this dis-

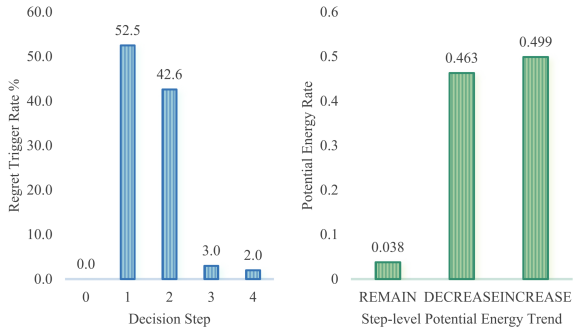


Figure 7: Analysis of regret trigger positions and potential energy dynamics: (a) Distribution of regret triggers across reasoning steps in CronQuestions; (b) Step-level potential energy trend on the MultiTQ.

tribution indicates a greater proportion of noisy or unstable decision-making steps in MultiTQ.

This is expected, as MultiTQ is inherently more complex—it contains longer reasoning chains, denser temporal dependencies, and generally lower model performance (Hit@1 = 0.543) compared with CronQuestions. As a result, the likelihood of retrieval errors or suboptimal decisions is higher, particularly in the early reasoning stages. The observed increase in potential energy for nearly half of the steps reflects this elevated noise level.

However, thanks to the design of our Regret Module and the calibrated threshold hyperparameter, the framework remains robust in identifying and correcting problematic steps. By tolerating minor energy fluctuations while triggering regret for more significant anomalies, our method successfully stabilizes the reasoning trajectory.

D.5 Extended Global Regret Analysis on MultiTQ

We replicate the global regret analysis on the MultiTQ dataset to assess the generality of our findings across temporally complex QA settings.

As shown in Table 12, the Regret Module (RM) is triggered in 59.9% of cases on MultiTQ, with a post-correction success rate of 27.5%. Similar to CronQuestions, the trigger rate is substantially higher on complex questions (78.8%) than on simple ones (52.9%), confirming that RM is more frequently activated when reasoning paths are prone to multi-step errors. For answer types, the trigger rate and correction success also vary: entity-based questions show a trigger rate of 57.8% with a success rate of 29.1%, while time-based questions achieve a higher trigger rate of 69.3%.

Overall, the regret trigger rate on MultiTQ (0.599) is significantly higher than that on CronQuestions (0.315), indicating that the more complex and multi-hop questions in MultiTQ are more likely to cause reasoning errors that activate the Regret Module. However, the post-correction accuracy on MultiTQ is much lower (0.275 vs. 0.709), suggesting that while the model frequently recognizes mistakes, it struggles to revise them effectively in more complex contexts.

D.6 Extended Efficiency Evaluation on MultiTQ

To evaluate the efficiency trade-off introduced by the Regret Stage, we compare Regret-Now with and without this module on the MultiTQ dataset, as shown in Table 13.

Since our potential energy computation relies on temporal embeddings with day-level granularity, we focus this analysis on **Single-granularity** questions—where both the question and answer are aligned at the day level. This ensures consistency in temporal resolution during the reasoning process and allows for a fair evaluation of the impact of the Regret Stage without interference from format mismatches in time representation.

Incorporating the Regret Stage leads to a notable improvement in final answer accuracy, rising from 0.617 to 0.693 (+7.6%). This demonstrates that explicitly revisiting and refining earlier decisions significantly enhances overall performance, particularly in the complex, multi-hop reasoning setting of MultiTQ.

However, this improvement comes at the cost of increased reasoning steps. The average number of steps rises from 2.73 to 3.25, reflecting the additional re-decision rounds triggered by the Regret Module. Interestingly, the *accuracy per step* metric slightly decreases from 0.226 to 0.213(-0.013). These results indicate that while the Regret Stage may slightly reduce local decision efficiency, it enhances global reasoning robustness and improves final outcomes.

E Case Study and Error Analysis

E.1 Case Study 1: Regret-Aware Correction in Temporal Reasoning

To further illustrate the effectiveness of our **Regret-Now** framework, we analyze a real example from our dataset:

> *"After 12 May 2014, which country did Iran make optimistic remarks about?"*

Rate	MultiTQ				
	Overall	Question Type		Answer Type	
		Simple	Complex	Entity	Time
RM Trigger Rate	0.599	0.529	0.788	0.578	0.693
Correct after RM	0.275	0.395	0.056	0.291	0.212

Table 12: Regret Module Trigger Rate and Post-correction Accuracy on the MultiTQ Dataset.

Model	Accuracy	Avg Steps	Accuracy per step
Regret-Now (w/o RS)	0.617	2.73	0.226
Regret-Now (ours)	0.693	3.25	0.213

Table 13: Efficiency Comparison of Regret-Now with and without Regret Stage on MultiTQ.

The correct reasoning trajectory is illustrated in the left of Figure 8, while the incorrect and regret-corrected trajectory is shown in the right of Figure 8.

The ideal reasoning path consists of three steps: (1) retrieving all entities to which Iran made optimistic comments, without any time constraints; (2) filtering for entities mentioned after the specified cutoff date (2014-05-12); and (3) selecting the earliest such entity—Kazakhstan on 2014-07-08—as the final answer. This trajectory corresponds to a monotonically decreasing potential energy curve: from -19.106 to -19.517, and finally to -22.559, as shown in the lower left of Figure 8. The smooth descent suggests increasing temporal coherence and logical plausibility throughout the reasoning process, with no regret triggered.

In contrast, the failed trajectory in the right results from an incorrect initial decision by the LLM. Instead of listing all possible optimistic remarks by Iran (Step 1), the model prematurely imposed a temporal constraint (`after 2014-05-12`), yielding only one internal remark that lacked sufficient candidates for downstream filtering. As a consequence, the Step 2 retrieval returned an empty set of entities.

This failure manifests physically as the virtual particle being infinitely distant from the subject particle in the TKG space, causing the potential energy to spike anomalously to 0. Such a discontinuity in the energy trend activates the **Regret Module**, which rolls back to the previous step. The model is then able to resume the correct trajectory and arrive at the correct answer.

E.2 Case Study 2: When the Potential Energy remains unchanged

While most reasoning steps exhibit a monotonic descent in potential energy, we observe that approximately 0.5% of transitions yield no change in energy. This phenomenon typically arises when the model retrieves the same virtual entity across consecutive steps. We now present a representative example to illustrate this behavior in detail.

> *"After 25 July 2014, which country signed formal agreements with Togo?"*

As shown in Figure 9, the model first invokes the operation `$get_head_entity` to retrieve all entities that signed formal agreements with Togo, regardless of time. This yields multiple candidates, among which `(China, 2015-05-01)` is the only one satisfying the temporal constraint posed in the question. In the next step, the model filters entities occurring strictly after `2014-07-25`, resulting in the exact same singleton: `(China, 2015-05-01)`.

Since both steps operate on identical virtual entities, the computed potential energy remains constant across Step 2 and Step 3. Specifically, both steps share the same relevance score $R_e = 7.965$, yielding an identical potential energy value of $\mathbf{-27.965}$. So, such plateaus do not indicate reasoning stagnation or error.

E.3 Error Analysis 1: Argument role confusion

To better illustrate the issue of argument role confusion, we present a representative failure case from our dataset:

> *"Who visited Guatemala before Susana Villaran?"* As shown in Figure 10

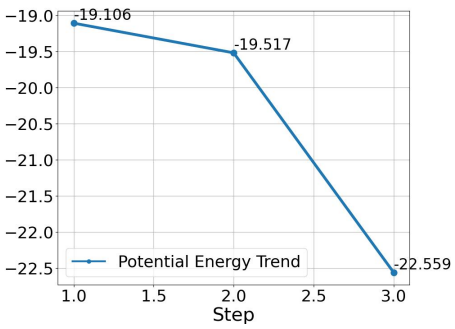
Question: After 12 May 2014, which country did Iran make optimistic remarks about?

Action 1: $get_tail_entity(Iran,Make_optimistic_comment,no time constraints)$
response 1: The get_tail_entity result is
Iran Make optimistic comment Iraq in 2005-06-15
Iran Make optimistic comment Mahmoud_Ahmadinejad in 2005-06-25
...
Iran Make optimistic comment Japan in 2015-09-21
entities = [('Iraq', '2005-06-15'), ('Mahmoud_Ahmadinejad', '2005-06-25'), '...', ('Japan', '2015-09-21')]
Re = -0.89403277635574340820312 5
Potential Energy = -19.106

Action 2: $get_after({entities},2014-05-12)$
response 2: The get_after result is
Kazakhstan in 2014-07-08
entities = [('Kazakhstan', '2014-07-08'), ('Head_of_Government_(Ukraine)', '2014-05-28'), ('South_Korea', '2015-09-13'), ('China', '2014-11-24'), ('China', '2015-02-27')]
Re = -0.4825232028961181640625
Potential Energy = -19.517

Action 3: $answer(Kazakhstan)$
Re = 2.558763027191162109375
Potential Energy = -22.559

Correct!



Action 1: $get_tail_entity(Iran,Make_optimistic_comment,2014-05-12)$
response 1: The get_tail_entity result is
Iran Make optimistic comment Member_of_Legislative_(Govt)_(Iran) in 2014-05-12
entities = [('Member_of_Legislative_(Govt)_(Iran)', '2014-05-12')]
Re = -3.02149222791194915771484375
Potential Energy = -16.979

Action 2: $get_after({entities},2014-05-12)$
response 2: The get_after result is [No results]
entities = []
Re = -∞
Potential Energy = 0

Action 2-Regret: $get_tail_entity(Iran,Make_optimistic_comment,no time constraints)$
response 2-Regret: The get_tail_entity result is
Iran Make optimistic comment Iraq in 2005-06-15
Iran Make optimistic comment Mahmoud_Ahmadinejad in 2005-06-25
...
Iran Make optimistic comment Japan in 2015-09-21
entities = [('Iraq', '2005-06-15'), ('Mahmoud_Ahmadinejad', '2005-06-25'), '...', ('Japan', '2015-09-21')]
r = -0.89403277635574340820312 5
Potential Energy = -19.106

Action 3: $get_after({entities},2014-05-12)$
response 3: The get_after result is
Kazakhstan in 2014-07-08
entities = [('Kazakhstan', '2014-07-08'), ('Head_of_Government_(Ukraine)', '2014-05-28'), ('South_Korea', '2015-09-13'), ('China', '2014-11-24'), ('China', '2015-02-27')]
r = -0.4825232028961181640625
Potential Energy = -19.517

Action 4: $answer(Kazakhstan)$
r = 2.558763027191162109375
Potential Energy = -22.559

Correct!

Figure 8: Case Study 1. Correct (Left) and Regret (Right) reasoning trajectory leading to the answer **Kazakhstan**, with steadily decreasing potential energy. The standard reasoning steps are highlighted in **yellow**, the erroneous step that causes the failure is marked in **red**, and the revised step issued by the Regret Module is marked in **green**

Question: After 25 July 2014, which country signed formal agreements with Togo?

Action 1: $get_head_entity(Togo,Sign_formal_agreement,no time constraints)$
response 1: The get_head_entity result is
Iran Sign formal agreement Togo in 2005-07-21
France Sign formal agreement Togo in 2005-08-03
...
China Sign formal agreement Togo in 2015-05-01
entities = [('Iran', '2005-07-21'), ('France', '2005-08-03'), '...', ('China', '2015-05-01')]
Re = 6.2496089935302734375
Potential Energy = -26.2496

Action 2: $get_after(entities,2014-07-25)$
response 2: The get_after result is
China in 2015-05-01
entities = [('China', '2015-05-01')]
Re = 7.965425014495849609375
Potential Energy = -27.965

Action 3: $answer(China)$
Re = 7.965425014495849609375
Potential Energy = -27.965

Figure 9: Case Study 2. An example where two consecutive reasoning steps retrieve the same entity (`China, 2015-05-01`), resulting in identical potential energy values.

The initial query in Operation 1 incorrectly treats *Susana Villaran* as the object of the visit rather than the subject. Specifically, the model issues the query `$get_time(Guatemala, Make_a_visit, Susana_Villaran)`, which mistakenly assumes that Guatemala visited Susana Villaran. This misinterpretation results in no retrieved records and a potential energy of 0.

The Regret Module successfully detects this anomaly and rewrites the query to `$get_time(Susana_Villaran, Make_a_visit, Guatemala)` in Operation 1-Regret, correctly identifying Susana Villaran as the visiting agent. This yields a valid result with a timestamp (2005-10-25) and a significant decrease in potential energy, indicating improved semantic alignment.

Subsequent steps proceed correctly: retrieving all entities who have visited Guatemala (Operation 3), filtering those who visited before 2005-10-25 Operation 4), and eventually identifying *Donald Rumsfeld* as the correct answer (Operation 5). Across this revised trajectory, the potential energy decreases step by step, confirming the improved reasoning stability.

This example highlights the importance of disambiguating subject-object roles in temporal relational queries. While our regret mechanism is able to resolve such confusions in many cases, this type of error remains frequent in failure cases—especially when entity names and syntactic structures make directionality ambiguous. Further improvements in role grounding or external relation labeling may help mitigate such errors more robustly.

E.4 Error Analysis 2: Suboptimal decision despite correct knowledge

To analyze whether incorrect reasoning stems from knowledge gaps or flawed decision-making, we inspect the token-level probability distribution returned by the OpenAI API via the `top_logprobs` field. This provides insight into how confident the model is in each candidate operation at a given step.

We present a representative failure case: *"After the International Monetary Fund, with which country did Japan first express its intention to negotiate?"* As shown in Figure 11, the model is expected to retrieve the timestamp of Japan's expression of intent to cooperate and compare it against that of the IMF. Therefore, the correct operation at this step is `get_time`.

However, instead of issuing `get_time(...)` to obtain the required temporal information, the

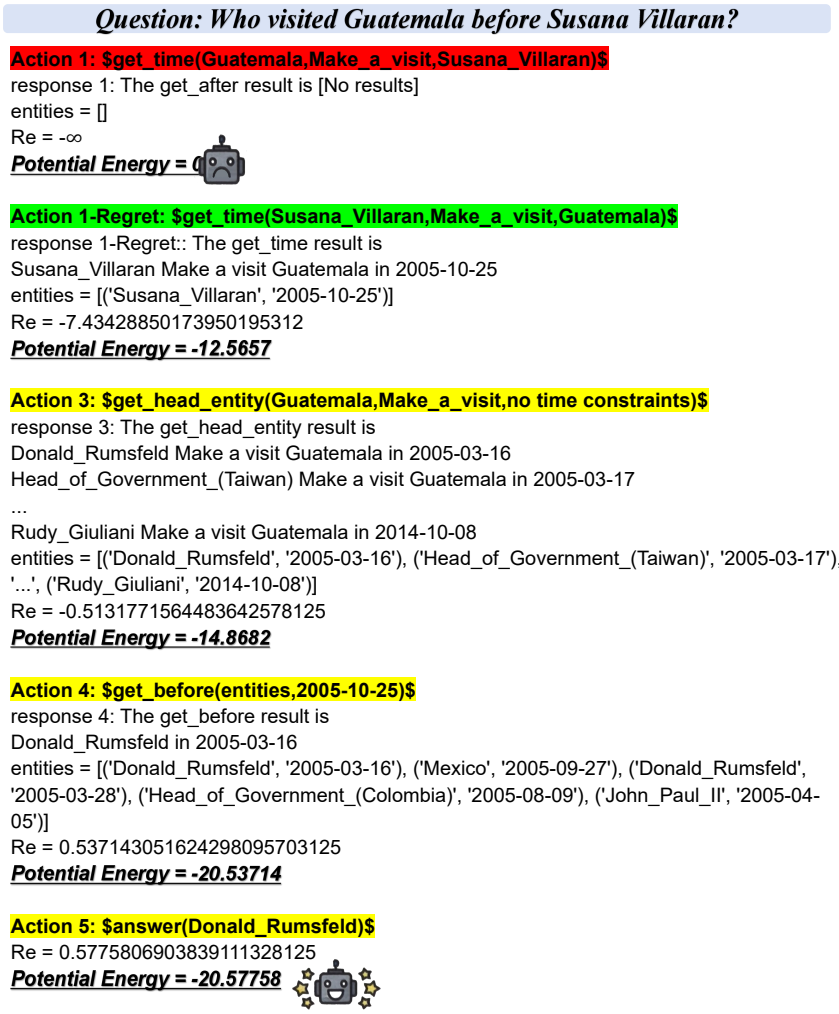


Figure 10: Error Analysis 1. Reasoning trajectory with argument role confusion. The initial query misidentifies the subject and object in the visit relation. Regret-Now revises the query direction and successfully recovers a stable reasoning path.

model chooses `get_tail_entity(...)`, an entity retrieval operation, which leads to an incorrect reasoning path.

From the `top_logprobs` list shown in the lower part of the figure, we observe that the token "_time" actually has the highest probability (logprob = -0.226), indicating that the model internally preferred `get_time`. Despite this, the model ultimately selected "_tail" (logprob = -1.72), leading to the execution of a suboptimal operation.

This case reveals that the model had access to the correct knowledge and internally recognized the correct direction, but still made a wrong decision.

F Prompt

F.1 LLM Decision Prompt

To guide the LLM through structured multi-step reasoning, we design a prompt template that introduces all available operation functions and constrains the model's output format. The full prompt used in our framework is shown in Table 14:

F.2 LLM Redecision Prompt

When the regret module detects a potential anomaly in the reasoning trace, it triggers the regret phase. During this phase, the LLM recalls an updated prompt that contains context related to the regret. The required prompts are listed in Table 15.

Question: After the International Monetary Fund, with which country did Japan first express its intention to negotiate?

Action 1:
$get_tail_entity(Japan,Express_intent_to_engage_in_diplomatic_cooperation_(such_as_policy_support),no time constraints)$
response 1: The get_tail_entity result is
Japan Express intent to engage in diplomatic cooperation (such as policy support) China in 2005-01-06
...
Japan Express intent to engage in diplomatic cooperation (such as policy support) Iran in 2015-12-17
entities = [('China', '2005-01-06'), ('China', '2005-01-08'), '...', ('Iran', '2015-12-17')]

Action 2:
$get_tail_entity(International_Monetary_Fund,Express_intent_to_engage_in_diplomatic_cooperation_(such_as_policy_support),no time constraints)$
response 2: The get_tail_entity result is
International_Monetary_Fund Express intent to engage in diplomatic cooperation (such as policy support) Romania in 2009-03-25
...
International_Monetary_Fund Express intent to engage in diplomatic cooperation (such as policy support) European_Central_Bank in 2015-06-02
entities = [('Romania', '2009-03-25'), ('Niger', '2010-10-15'), '...', ('European_Central_Bank', '2015-06-02')]

top_logprobs = [{'token': '_time', 'logprob': -0.22622278, 'bytes': [95, 116, 105, 109, 101]},
{'token': '_tail', 'logprob': -1.7262228, 'bytes': [95, 116, 97, 105, 108]},
{'token': '_first', 'logprob': -3.9762228, 'bytes': [95, 102, 105, 114, 115, 116]},
{'token': '_last', 'logprob': -5.601223, 'bytes': [95, 108, 97, 115, 116]},
{'token': '_before', 'logprob': -6.476223, 'bytes': [95, 98, 101, 102, 111, 114, 101]}]

Figure 11: Error Analysis 2. Suboptimal decision case. The model selects the second-most likely operation (`get_tail`) instead of the correct one (`get_time`), despite having higher confidence in the latter.

F.3 LLM System Initialization Prompt

Before reasoning begins, we initialize the LLM using a dedicated system prompt that defines its role, available tools, and expected behavior. This initialization prompt ensures consistency across all reasoning sessions, as shown in Table 16.

G Experimental Environment

All experiments are conducted on a physical NVIDIA RTX 3090 GPU with 24GB memory. The RTX 3090 server provides 14 vCPUs (Intel Xeon Gold 6330 @ 2.00GHz). The software environment includes PyTorch 1.10.0, Python 3.8 (Ubuntu 20.04), and CUDA 11.3.

Prompt of LLM Decision Making

Please use the tool provided below to interact with the knowledge graph. You will find a list of operations categorized into time-based queries, entity queries, and specific time queries. There may be more than one answer to the question, but you only need to answer one correct answer that satisfies the question.
To solve this question, you need to first identify the entities and relationships in the question, selecting the appropriate operations to retrieve the required information, and finally, providing the correct answer.

Time-based Queries:

Retrieve the time of a specific event using $get_time(HEAD, RELATION, TAIL)$.

Find events before a time using $get_before(ENTITY_LIST, TIME)$.

Find events after a time using $get_after(ENTITY_LIST, TIME)$.

Filter events between two times using $get_between(ENTITY_LIST, START, END)$.

Entity Queries:

Get tail entity using $get_tail_entity(HEAD, RELATION, TIME)$.

Get head entity using $get_head_entity(TAIL, RELATION, TIME)$.

Specific Time Queries:

Use $get_first(ENTITY_LIST)$ to find the earliest.

Use $get_last(ENTITY_LIST)$ to find the latest.

Use $answer(YOUR_ANSWER)$ to submit your final answer.
Examples for your reference: {examples}

(end of examples)

Current Challenge:

Question: {question}

Previous Operations: {history}

(end of previous operations)

Available Operations: {operations}
Choose your next operation from the available operations above.

Operation:

$[Your selected operation]$

Reason:

[Explain the reason for choosing this operation]

Table 14: Prompt used for LLM step-wise decision making, including available operations and output format.

Prompt of Regret-based LLM Decision Making

Please note that you have answered the following questions before, but there was an error.

Your wrong choice last time was + **last_wrong_operation**. Please read carefully and answer again.

In particular, please pay attention to the following: maintain strict logical consistency; ensure that all required knowledge has been retrieved before invoking time-based operations such as get_before or get_after; and carefully consider the temporal order between before and after.

Please use the tool provided below to interact with the knowledge graph. You will find a list of operations categorized into time-based queries, entity queries, and specific time queries. There may be more than one answer to the question, but you only need to answer one correct answer that satisfies the question.
To solve this question, you need to first identify the entities and relationships in the question, selecting the appropriate operations to retrieve the required information, and finally, providing the correct answer.

Time-based Queries:

Retrieve the time of a specific event using $get_time(HEAD, RELATION, TAIL)$.

Find events before a time using $get_before(ENTITY_LIST, TIME)$.

Find events after a time using $get_after(ENTITY_LIST, TIME)$.

Filter events between two times using $get_between(ENTITY_LIST, START, END)$.

Entity Queries:

Get tail entity using $get_tail_entity(HEAD, RELATION, TIME?)$.

Get head entity using $get_head_entity(TAIL, RELATION, TIME?)$.

Specific Time Queries:

Use $get_first(ENTITY_LIST)$ to find the earliest.

Use $get_last(ENTITY_LIST)$ to find the latest.

Use $answer(YOUR_ANSWER)$ to submit your final answer.
Examples for your reference: {examples}

(end of examples)

Current Challenge:

Question: {question}

Methodology: {methodology}

(end of methodology)

Previous operations: {history}

(end of previous operations)

Available operations: {operations}
Choose your next operations from the available operations above.

Operation:

$[Your selected operation]$

Reason:

[Explain the reason for choosing this operation]

Table 15: Regret-enhanced prompt for LLM step-wise decision making. The model is informed of its previous error and asked to revise its decision with better reasoning.

Prompt of System Role Description
You are a decision-maker guiding an agent through a temporal knowledge graph. The goal is to help the agent navigate and query until it uncovers the correct answer. The agent will suggest various methods and queries, and based on the question's semantics and previous steps, you'll choose the best option. Let's do step by step.

Table 16: System role prompt used when initializing the LLM, providing high-level guidance for sequential temporal reasoning.