Grounding LLM Reasoning with Knowledge Graphs

Anonymous ACL submission

Abstract

001 Knowledge Graphs (KGs) are valuable for representing relationships between entities in a 003 structured format. Traditionally, these knowledge bases are queried to extract specific information. However, question-answering (QA) over KGs poses a challenge due to the intrinsic complexity of natural language compared to 007 800 the structured format and the vast size of these graphs. Despite these challenges, the structured nature of KGs offer a robust foundation for grounding the outputs of Large Language Models (LLMs), enhancing reliability and control for organizations.

In this work, we introduce a novel integration of reasoning strategies with KGs, anchoring each step or "thought" of the reasoning chains in KG data. This approach uses recent advancements in LLMs, applying reasoning methods during inference to improve performance and capabilities. We evaluate both agentic and automated search methods across several reasoning strategies, including Chain-of-Thought (CoT), Tree-of-Thought (ToT), and Graph-of-Thought (GoT), using GRBench, a benchmark dataset for graph reasoning with domainspecific graphs. Our experiments demonstrate that this innovative approach achieves a significant performance improvement of at least 26.5% over baseline models, highlighting the benefits of grounding LLM reasoning processes in structured KG data.

1 Introduction

017

018

026

027

032

Large Language Models (LLMs) have shown remarkable versatility in answering questions posed in natural language. This is mainly due to their ability to generate text, their broad internal knowledge, and their capacity to access external information (Zhuang et al., 2023; Lewis et al., 2020). However, a significant area for improvement is their tendency to produce information that, while plausiblesounding, is often unverifiable and lacks traceable origins and sources (Tonmoy et al., 2024). The LLM generation process heavily relies on their internal parameters, making it difficult to link their outputs to external sources (Ko et al., 2024; Zheng et al., 2024). This limitation challenges their reliability in industrial applications (Luo and Specia, 2024). In applied settings, where LLMs handle critical operations, integrating them with domainspecific knowledge is essential. Fine-tuning LLMs for new domains is labor-intensive, especially for companies with proprietary data facing privacy and legal issues. Therefore, developing processes to effectively connect LLMs with external knowledge bases is crucial. 043

044

045

046

047

050

051

052

053

056

057

059

060

061

062

063

064

065

067

068

069

071

072

073

074

075

076

077

078

079

081

When connecting LLMs with external knowledge, techniques such as Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) or SQLcode generation (Rajkumar et al., 2022) can be employed. However, they often fail to capture the dynamic relationships between concepts that are necessary for comprehensive understanding. These approaches typically assume that knowledge is wellrepresented in discrete units, such as documents or tables, which can lead to incomplete insights when dealing with interconnected knowledge that spans multiple sources.

KGs effectively represent complex knowledge by organizing real-world entities and their relationships. Using KGs to guide LLM reasoning is a novel approach recently explored (Luo et al., 2024). In this work, we introduce a novel integration of reasoning strategies with domain-specific KGs, (academic, literature, healthcare, etc.) from GRBench dataset (Jin et al., 2024), enabling a more comprehensive understanding of complex queries. Unlike previous methods, our approach systematically connects each reasoning step with the KG, leveraging its interconnected nature to enhance the reasoning capabilities of LLMs. Each sample in GRBench contains a manually crafted question and its corresponding answer, which can be obtained by retrieving relevant information from the



Figure 1: Methods for Question-Answering in KGs, explaining the general framework described in Section 4. Left: *Agent*. LLM decides to take one of the predefined actions to connect with the graph. **Right:** *Automatic Graph Exploration*. Entities are extracted in each reasoning step, triggering a search for each identified entity.

graph as context. We present various reasoning methods that connect every *thought* or *step* with the KG. These methods include Chain-of-Thought (CoT – Wei et al. (2022)), Tree-of-Thought (ToT– Yao et al. (2024)) and Graph-of-Thought (GoT – Besta et al. (2024)). The strategies are shown in Figure 2. Our results demonstrate significant improvements in generating accurate answers from the graph, achieving state-of-the-art performance on GRBench.

090

101

102

104

105

106

108

109

110

111

112

Our contributions can be summarized as follows:

- We present a versatile framework that integrates reasoning strategies with KG search during inference. This framework is designed to easily incorporate additional reasoning strategies, making it adaptable to any domain and application.
- Our results achieve state-of-the-art performance on GRBench, demonstrating significant average performance improvement of at least 26.5% over the CoT method.
- We conduct an analysis of various aspects of our methods, including the impact of the steps, tree width and model sizes (Section 7).
- By systematically connecting each reasoning step with the KG, our approach improves the interpretability of LLM outputs, ensuring that responses are not only accurate but also traceable to their knowledge sources.

2 Related Work

LLMs require large amounts of data and resources for training (Villalobos et al., 2022). The need to leverage these models with external data after they are trained has driven the popularity of RAG methods, which incorporate external data (Lewis et al., 2020; Khattab et al., 2022). Recent advances have combined RAG with structured knowledge such as ontologies, enhancing LLM reasoning capabilities (Li et al., 2024). Our work introduces a framework that employs advanced reasoning strategies, grounding LLM outputs in domain-specific KGs for improved performance in specific domains. This approach facilitates more targeted and iterative interactions with knowledge graphs, distinguishing it from traditional RAG methods. 113

114

115

116

117

118

119

120

121

122

123

124

125

127

128

129

130

131

132

133

134

136

137

138

139

140

Structured Knowledge Structured knowledge, such as Databases or KGs, provides organizations with reliable sources of information that can be more easily maintained and automatically updated. KGs, in particular, offer an adaptable knowledge model that captures complex relationships between interconnected concepts. Some research has focused on developing models that can interact with multiple types of structured knowledge: StructLM (Zhuang et al., 2024), and examining the impact of incorporating structured knowledge into the pretraining of LLMs (Moiseev et al., 2022).

Integrating KGs with LLMs (Peng et al., 2024) 141 The integration of KGs with LLMs has emerged 142 as a promising approach to enhance AI systems' 143 reasoning capabilities and reliability. In general, 144 we distinguish four primary methods for enabling 145 LLMs to interact with graphs: (1) Learning graph 146 representations (Fatemi et al., 2024; Chai et al., 147 2023), however these latent representations cur-148 rently fall short of text-based methods on Knowl-149 edge Graph Question Answering (KGQA) tasks. 150 (2) Using Graph Neural Network (GNN) retrievers 151 to extract relevant entities and provide text-based 152 input to the model (He et al., 2024; Mavromatis 153 and Karypis, 2024). (3) Generating code, such as 154 SPARQL, to retrieve data from the graph (Li et al., 155 2023). Finally, (4) Methods that allow step-by-step 156 interaction with KG (Luo et al., 2024; Yu et al., 157 2022; Luo et al., 2023). These last methods cur-158 rently perform best on KGQA tasks. 159

LLM Reasoning with Graphs There is a grow-160 ing interest in leveraging KGs to enhance the rea-161 soning capabilities of LLMs. KGs not only serve 162 as a structured data source but also provide a frame-163 work for understanding and improving the reason-164 ing processes of LLMs (Wang et al., 2024). This in-165 tegration enables models to generate more coherent 166 and contextually relevant responses while allowing the tracing and verification of the reasoning steps. 168 The most effective methods typically involve a step-169 by-step interaction between LLMs and graphs, as 170 discussed in the previous paragraph. Notable ex-171 amples of this approach include the works of Sun 172 et al. (2024); Wen et al. (2024); Luo et al. (2024); 173 Kim et al. (2023); Li et al. (2025). Recent research, 174 including our own, has begun to explore the in-175 tegration of traditional LLM reasoning strategies 176 with KGs, as shown in studies by Jin et al. (2024); 177 Markowitz et al. (2024). 178

3 Background

179

181

182

183

185

186

187

189

In this section, we formalize the prerequisite knowledge relevant to this paper. We use p_{θ} to denote a pre-trained language model with parameters θ . And letters x, y, z to refer to a language sequence. $x = (x_1, x_2, ..., x_n)$, where each is x_i is a such that $p_{\theta}(x) = \prod_{i=1}^{n} p_{\theta}(x_i | x_{1...i-1})$.

Knowledge Graphs (KGs) A KG is a heterogeneous directed graph that contains factual knowledge to model structured information. In a KG, nodes represent entities, events, or concepts, while edges represent the connection and types of relations between them. Formally, a KG is represented as \mathcal{G} , defined by a set of triples $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where \mathcal{E} , \mathcal{R} denote the set of entities and relations, respectively.

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

224

225

226

227

228

229

230

231

232

233

234

235

236

237

Knowledge Graph Question-Answering (KGQA) It is a reasoning task that leverages KGs. Given a natural language question, q, and an associated KG, \mathcal{G} , the goal is to develop a method that retrieves the correct answer, a, based on the knowledge extracted from the KG: $a = f(q, \mathcal{G})$.

Step-by-step Reasoning with LLMs To improve the reasoning capabilities of LLMs at inference time, a common approach is to generate intermediate reasoning steps. The key idea is the introduction of intermediate steps, $Z_{p\theta} = z_1, ..., z_n$, to add inference sources to bridge the q and a. This approach enables models to break down complex, multi-step problems into smaller, manageable intermediate steps. By doing so, additional computational resources can be allocated to problems that necessitate more reasoning steps.

4 Method

This work aims to show how conditioning the model at each step of the LLM reasoning process can improve domain-specific question answering based on graphs. Our method combines reasoning strategies for LLMs: *CoT, ToT, GoT* with 2 graph interaction methods: (1) Agent, an agent to navigate the graph; and (2) Automatic Graph Exploration, an automatic graph traversal mechanism based on the generated text.

4.1 Reasoning Strategies

Chain-of-Thought (CoT) CoT (Wei et al., 2022) is a well-known reasoning method that involves generating a sequence of logical steps, where each step builds upon the previous ones, ultimately leading to a conclusion. Formally, it generates a sequence of reasoning steps $Z_{p_{\theta}}(q) =$ $\{z_1, z_2, \ldots, z_n\}$, where each step z_i is sampled sequentially given the input query q, all previous steps and and graph information from all steps , \mathcal{G}' , as $z_i \sim p_{\theta}^{\text{CoT}}(z_i|q, \mathcal{G}', z_{1...i-1})$. The final answer a is derived from this reasoning process given all the generated thoughts $a \sim p_{\theta}^{\text{CoT}}(a|q, \mathcal{G}', z_{1...n})$. In practice, it is sample as a continuous language sequence.Figure 2a represents this method, where each step is linked to the KG.



Figure 2: Reasoning Strategies: This figure illustrates different LLM reasoning strategies to navigate the potential answer space: CoT, ToT, GoT. Each strategy consists of "thoughts" connected to the Knowledge Graph (KG) through search methods, as detailed in Section 4.2: Agent or Graph Exploration.

Tree-of-Thought (ToT) ToT (Yao et al., 2024) generalizes CoT by modeling the reasoning process as a tree, enabling simultaneous exploration of multiple reasoning paths. Starting from an initial state $s_0 = [q]$, where q is the input, ToT incrementally expands each state by generating multiple candidate thoughts:

239

240

241

242

243

244

246

247

248

251

253

260

261

263

264

$$z_{i+1}^{(j)} \sim p_{\theta}(z_{i+1} \mid s_i), \quad j = 1, \dots, k$$
 (1)

Each candidate thought represents a node in the tree, forming new states. These states are evaluated by a heuristic scoring function $V(p_{\theta}, s)$, guiding the selection and pruning of branches. Search strategies, such as breadth-first search (BFS), systematically explore this tree:

$$S_t = \operatorname{argmax}_{S' \subseteq \hat{S}_t, |S'| = b} \sum_{s \in S'} V(p_\theta, s) \quad (2)$$

where \hat{S}_t denotes candidate states at step t, and b limits the breadth. We implement two versions of heuristic functions V to select the top t states:

- 1. Selection: The LLM directly chooses the top *t* states to proceed, discarding the others.
- 2. Score: The states are ranked by a heuristic voting mechanism: $V(p_{\theta}, S)(s) = \mathbb{P}[s = s^*]$ where the LLM is prompted to estimate probability of the current state solving the given input question.

This structured search and pruning strategy significantly enhances the model's ability to solve complex reasoning tasks requiring strategic planning. **Graph-of-Thought (GoT)** GoT (Besta et al., 2024) extends ToT by organizing reasoning into a directed graph structure G = (V, E), where each node represents a thought and edges reflect dependencies. Starting from an initial thought, new thoughts are generated similarly to ToT and added to the graph. Each new thought is connected to its parent, and additional reasoning chains can be formed through merging operations:

$$z_{i+1} = A(z_i^{(a)}, z_i^{(b)})$$
(3)

267

268

269

270

271

272

273

274

275

276

277

278

281

282

283

285

287

290

291

293

294

295

where A denotes a merge operation that integrates two thought chains into a single coherent reasoning step. The merged thought is added as a new node with edges from both parents.

In our implementation, thoughts are evaluated using either a Selection or Score-based strategy as in ToT. Merged thoughts inherit information from both parents and can enhance robustness. At each depth, a fixed number of thoughts are retained using breadth-first traversal and evaluated for progression. This architecture should enable dynamic refinement and fusion of ideas, offering a generalization of CoT and ToT for knowledge-grounded question answering.

4.2 LLM + KG Interaction Methods

We implement methods to connect reasoning strategies with KGs. The LLM interacts with the KG at every step. This retrieves new information and conditions the model for subsequent steps. We present 2 methods to achieve this interaction, both illustrated in Appendix B.

4.2.1 Agent

298

315

319

321

323

325

326

327

334

338

340

342

344

This approach creates an agent that interacts with the graph, following the methodology initially described in ReACT (Yao et al., 2022). After generating a thought, the LLM selects from a set of actions based on the given thought. Each step in the reasoning chain consists of an interleaved sequence: *thought* \rightarrow *action* \rightarrow *retrieved data*. This method implements four actions as described in GraphCoT (Jin et al., 2024):

RetrieveNode(Text) Identifies the related nodein the graph using semantic search.

310 NodeFeature(NodeID, FeatureName) Re311 trieves textual information for a specific node from
312 the graph.

NeighborCheck(NodeID, EdgeType) Retrieves
neighbors' information for a specific node.

NodeDegree(NodeID, EdgeType) Returns the degree (#neighbors) for a given node and edge type

These actions collectively enable the agent to navigate and extract meaningful information from the graph, enhancing the reasoning capabilities of the LLM by grounding its thoughts in structured, retrievable data.

4.2.2 Automatic Graph Exploration

This method incrementally searches the graph by interleaving language generation with structured retrieval. At each step, the LLM generates a new "thought" based on previous thoughts and retrieved triples. Entities mentioned in the generated text are automatically extracted using LLM prompts and serve as anchors for further graph exploration.

Graph exploration proceeds through a multi-step Search + Prune pipeline, inspired by the process described in Sun et al. (2024). For each unvisited entity, the system first retrieves and prunes relation types using LLM guidance. Then, for each selected relation, neighboring entities are discovered and filtered using a second round of pruning. The model selects only the most relevant neighbors based on their contextual fit with the question and previous reasoning steps. This hierarchical pruning - first on relations, then on entities - ensures the method remains computationally tractable while preserving interpretability. The overall traversal follows a breadth-first search (BFS) pattern, with pruning decisions at each level directed by LLM. This process is shown in Figure 3.



Figure 3: Automatic Graph Exploration: It begins by extracting entities from text (query/thought), followed by selecting relevant relations and neighbors with the LLM. The resulting entity-relation-entity combinations form triples used to expand the reasoning chain.

This iterative reasoning and retrieval process allows the model to dynamically condition future steps on progressively more relevant subgraphs. Unlike agentic methods that rely on predefined actions, the automatic approach operates in the graph space guided by the natural language, providing more freedom in the generation. The mechanism is designed to maximize information gain at each step while avoiding graph overgrowth. More details are provided in Algorithm 1. 347

348

349

350

351

352

353

354

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

5 Experiments

Benchmark We use the GRBrench dataset (Jin et al., 2024) to evaluate our methods. This dataset is specifically designed to evaluate how effectively LLMs can interact with domain-specific graphs to solve the given problem. It includes several graphs spanning various general domains. For our evaluation, we selected 7 graphs across multiple domains, excluding those with excessively high RAM requirements that exceed our available resources. Details the graph statistics and can be found in Appendix A.

Baselines The proposed methods, *Agent* and *Automatic Graph Exploration*, applied to CoT, ToT, and GoT, are compared against the following baseline methods: (1) **Zero-Shot**: Directly querying the model to answer the question without additional context. (2) **Text RAG** (Gao et al., 2023): Textretrieval method that uses the text representation of nodes as input for the query, with the retrieved data serving as context for the model. (3) **Graph RAG**: Includes node neighbors (1-hop) for additional context beyond Text RAG. (4) **Graph CoT** (**Agent**): Implements Graph CoT (Jin et al., 2024) as an agent for CoT reasoning, utilizing the actions described in Section 4.2.

Experimental methods We implement the methods described in Section 4, extending (1) Agent 384 and (2) Automatic Graph Exploration with various reasoning strategies during inference: (1) CoT, (2) ToT, and (3) GoT. For the latter two, we include their State Evaluation methods at each step: (1) Selection and (2) Score. In the results presented in Table 1, we set n = 10 steps for all methods. ToT and GoT use a branching factor and Selection of k = t = 3. Our experiments use only openaccess Llama 3.1 (Instruct) (Dubey et al., 2024) as the backend models, which enhances reproducibility and allows for unlimited free calls. Specifically, we employ the 8B, 70B, and 405B versions, using the FP8 variant for the 405B model.

Evaluation We use rule-based and model-based metrics to evaluate the models, following GRBench paper (Jin et al., 2024). For the rule-based metric, we use Rouge-L (R-L) (Lin, 2004), which measures the longest sequence of words appearing in same order in both generated text and ground truth answer. For model-based metric, we prompt GPT-40 to assess if the model's output matches ground truth answer. GPT4Score is percentage of answers that GPT-40 identifies as correct.

Implementation Details The experiments are run on NVIDIA TITAN RTX or NVIDIA A100 using Python 3.8. The models are deployed with vLLM (Kwon et al., 2023), a memory-efficient library for LLM inference and serving. For the baseline, Mpnet-v2 is used as the retriever, and FAISS (Johnson et al., 2019) is employed for indexing.

6 Results

398

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

The main results from both the baselines and ex-416 perimental methods, evaluated using R-L, are pre-417 sented in Table 1. For brevity, additional results 418 using GPT4Score can be found in Appendix D. We 419 highlight three key insights from the findings: (1) 420 The agentic method generally outperformed auto-421 matic graph exploration, indicating that targeted 422 interactions with the KG enhance answer accuracy. 423 (2) The ToT strategy demonstrated superior perfor-424 mance by effectively exploring multiple reasoning 425 paths. (3) Although GoT strategy showed potential, 426 it did not significantly outperform ToT, suggesting 427 a need for further refinement in merging divergent 428 reasoning results. These results show the impor-429 tance of reasoning strategies to navigate multiple 430 paths in the graph. 431

Agent vs Graph Search In our experimental re-432 sults, the agentic method outperformed graph ex-433 ploration approach across most datasets and rea-434 soning strategies. The agent-based method, which 435 involves LLM selecting specific actions to interact 436 with KG, consistently improves performance as the 437 number of reasoning steps increases, as shown in 438 Section 7. This suggests that while graph explo-439 ration can quickly provide relevant information, the 440 agentic method's iterative and targeted interactions 441 with KG yield more accurate and comprehensive 442 answers over longer sequence of steps. 443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

Tree of Thought (ToT) The ToT reasoning strategy showed superior performance across its various interaction methods and state evaluators, as summarized in Table 1. ToT achieved performance improvements of 54.74% in agent performance and 11.74% in exploration mode compared to the CoT version. However, this improvement comes with the trade-off of increased inference time, highlighting the effectiveness of inference-time strategies. In the ToT approach, a state evaluator is used to select the most promising branch for the next step. We also compared the two State Evaluation methods: Selection and Score.

Graph of Thought (GoT) The results for GoT strategy are summarized in Table 2. Due to additional computational time required, we report results for two datasets only. The GoT strategy did not outperform ToT approach. Our initial hypothesis was that LLMs could effectively integrate divergent results from different branches. However, in practice, the models struggled to merge these results effectively. Specifically, in case of Graph-Search, models often failed to effectively combine different triples found in separate branches. This finding presents an interesting area for future research, potentially leading to development of more sophisticated reasoning strategies for merging results from different branches.

7 Analysis & Ablation studies

In this section, we want to better understand the nuances of our methods for LLM and KG grounding. We conduct an analysis on the Academic datasets from the benchmark, as they all contain the same number of samples and feature questions generated from similar templates to ensure a controlled comparison.

	Method		Model	Healthcare	Goodreads	Biology	Chemistry	Materials Science	Medicine	Physics
	Base		Llama 3.1 8B-Ins	7.32	6.18	10.68	11.69	8.95	8.77	11.52
			Llama 3.1 70B-Ins	9.74	9.79	11.49	12.58	10.40	12.21	12.61
			Llama 3.1 405B-Ins	8.66	12.49	10.52	13.51	11.73	11.82	11.63
nes	Text-RAG		Llama 3.1 8B-Ins	8.24	14.69	12.43	11.42	9.46	10.75	11.29
seli			Llama 3.1 70B-Ins	10.32	18.81	11.87	16.35	12.25	12.77	12.54
Ba			Llama 3.1 405B-Ins	11.61	16.23	16.11	13.82	14.23	15.16	16.32
			Llama 3.1 8B-Ins	12.94	22.30	30.72	34.46	30.20	25.81	33.49
	Graph-RAG		Llama 3.1 70B-Ins	17.95	25.36	38.88	40.90	41.09	31.43	39.75
			Llama 3.1 405B-Ins	16.12	23.13	37.57	42.58	37.74	33.34	40.98
			Llama 3.1 8B-Ins	16.83	30.91	20.15	18.43	26.29	14.95	21.41
_	Agent		Llama 3.1 70B-Ins	33.48	40.98	50.00	51.53	49.6	48.27	44.35
oT			Llama 3.1 405B-Ins	28.41	36.56	41.35	48.36	47.81	42.54	35.24
θũ	Graph Explore		Llama 3.1 8B-Ins	25.58	32.34	36.65	35.33	31.06	31.05	35.96
			Llama 3.1 70B-Ins	29.41	29.60	44.63	49.49	39.23	38.87	45.52
			Llama 3.1 405B-Ins	28.45	43.06	36.93	38.71	47.49	55.66	32.73
			Llama 3.1 8B-Ins	28.91	52.25	43.81	44.18	43.49	36.07	39.56
	Score	Score	Llama 3.1 70B-Ins	38.51	51.58	64.44	61.13	55.19	63.00	55.33
			Llama 3.1 405B-Ins	47.51	50.73	70.34	64.9	49.02	65.40	44.63
	Agent		Llama 3.1 8B-Ins	28.67	50.59	42.33	37.07	40.81	33.17	36.50
_		Select	Llama 3.1 70B-Ins	40.26	52.59	64.53	66.84	61.42	61.21	55.89
aph			Llama 3.1 405B-Ins	46.90	51.68	70.27	67.95	63.74	64.23	59.56
ĞΡ			Llama 3.1 8B-Ins	24.49	36.80	35.81	36.41	34.28	34.49	37.69
		Score	Llama 3.1 70B-Ins	32.79	38.19	53.83	58.25	48.55	52.18	48.07
	Graph		Llama 3.1 405B-Ins	33.90	42.68	46.87	57.43	50.46	55.56	48.73
	Explore		Llama 3.1 8B-Ins	25.04	37.8	36.34	38.5	32.44	33.31	34.85
		Select	Llama 3.1 70B-Ins	33.40	39.13	54.78	58.53	47.19	51.13	47.51
			Llama 3.1 405B-Ins	33.82	43.63	44.47	59.06	48.52	55.62	46.07

Table 1: Rouge-L (R-L) performance results on GRBench (Jin et al., 2024), comparing standard LLMs, Text-RAG, Graph-RAG, Graph-CoT, and Graph-ToT. Experiments are described in Section 5, using LLama 3.1 - Instruct backbone models with sizes 8B, 70B, and 405B-FP8.

Method		Model	Healthcare	Biology
	e	Llama 3.1 8B-Ins	29.11	33.25
	CO	Llama 3.1 70B-Ins	30.88	56.64
ent	S	Llama 3.1 405B-Ins	43.53	48.1
Ag.	ct	Llama 3.1 8B-Ins	29.05	40.37
	ele	Llama 3.1 70B-Ins	40.74	65.59
	S	Llama 3.1 405B-Ins	47.63	71.49
e	e	Llama 3.1 8B-Ins	24.96	21.72
lor	S Llan	Llama 3.1 70B-Ins	31.24	50.70
ExI		Llama 3.1 405B-Ins	35.00	39.10
[hd	ct	Llama 3.1 8B-Ins	25.06	21.84
jraj	ele	Llama 3.1 70B-Ins	36.95	52.32
	S	Llama 3.1 405B-Ins	33.74	54.64

Table 2: Graph-GoT results on GRBench using Rouge-L with Llama 3.1 Instruct sizes 8B, 70B, and 405B.

How does the number of steps affect the results? Figure 4 illustrates the effect of varying the number of steps in the KG interaction methods (Agent, Explore) across all academic datasets. The plots indicate that graph exploration performs better with fewer steps, as it automatically traverses the graph for the identified anchor entities. Conversely, the agentic methods improve as the number of steps increases, eventually achieving better performance. This highlights a key trade-off between passive retrieval through exploration and more deliberate reasoning via agentic planning.

480

481

482

483

484

485

486

487

488

489

490

491



Figure 4: Effect of the number of steps in the LLM-KG Interaction Methods. The Agent requires more steps to obtain the performance of the Graph Exploration, while the Graph Exploration only needs the anchor entities to perform the search within the graph.

What is the effect of Search Depth in Automatic Graph Exploration? We analyze the effect of search depth in Figure 5, which presents performance results across various depths, with fixed step size of one. The results demonstrate that the performance of depth-first search plateaus at depth of 3, highlighting the relevance of search exploration with respect to the given query. Beyond this point, deeper traversal yields no significant gains, likely due to diminishing relevance of distant nodes.

497

498

499

500

501



Figure 5: Effect of the Search depth in Graph Exploration interaction method for a fixed steps number. The method can achieve relatively good performance with the anchor entities extracted from the question.



Figure 6: Impact of tree width on Agentic ToT performance. It shows a general trend of performance improvement with increasing tree width.

What is the effect of tree width in the reasoning strategy (ToT)? Based on experimental results across all academic datasets, we observe performance variations among different methods. To gain further insight, we analyze the effect of tree width on results, as shown in Figure 6. We notice a slight upward trend in performance as the tree width increases, although the difference is more pronounced between CoT and ToT itself, going from one branch to two. The added computational time and resources likely contribute to this performance enhancement.

502

503

504

505

507

509

510

511

512

513

What is the influence of the state evaluator? 514 Figure 7 explores the impact of state evaluators, 515 specifically Score and Select, within the ToT frame-516 work. The analysis indicates that, while there is no 517 significant difference between the two methods, the 518 Select evaluator generally yields slightly better re-519 sults. This trend is especially evident in the context 520 of the Agent's performance, though the advantage 521 is less pronounced in automatic graph exploration.



Figure 7: Influence of the State Evaluators in ToT. The Select method obtains better results over Score method.

How are errors different for each strategy? To understand failure patterns, we define three error types: (1) Reached limit — the reasoning hit the step limit; (2) Answer found but not returned the correct answer appeared but was not output; (3) Wrong reasoning step — the model followed an illogical step. Using GPT-40, we labeled a larger set of answers and traces. As shown in Figure 8, ToT and GoT show more "answer found but not returned" cases than CoT, suggesting better retrieval but occasional failures in synthesis. This comes with a slight rise in logical errors, likely due to the complexity of multiple reasoning paths.



Figure 8: Error distribution across reasoning strategies. ToT and GoT reduce unanswered cases but slightly increase logical errors due to more complex reasoning.

8 Conclusion

We present a framework for grounding LLM reasoning in KGs by integrating each step with structured graph retrieval. By combining strategies like CoT, ToT, and GoT with adaptive graph search, our method achieves state-of-the-art performance on GRBench. The approach enables inference-time reasoning, offering flexibility across domains, and laying the groundwork for future advances in structured knowledge integration. 536

537

538

539

540

541

542

543

544

9 Limitations

546

549

550

551

556

557

561

562

563

566

569

570

573

574

578

579

583

584

585

589

591

592

In this work, we demonstrate how LLMs can be used to explore a graph while conditioning the next steps based on the graph's results. We show that the two approaches presented achieve superior results in graph exploration. Integrating KGs with LLMs can provide complex relational knowledge for LLMs to leverage. However, the system's performance will depend on the knowledge encoded in the graph and the models' capabilities.

Extending inference-time reasoning methods for LLMs is significantly constrained by computational resources and the time available to the user. We analyze the computational complexity of the methods in Appendix E, where we show the exponential growth of ToT and GoT due to its branching structure. GoT further compounds this by allowing merges between reasoning paths, which increases the total number of evaluations. Additionally, this work is constrained by the computational resources needed to load the graph into memory, as it demands a significant amount of available RAM.

While LLMs conditioned on external knowledge can generate outputs based on accessed content, their generated output is not strictly limited to that information. Thus, they may still generate hallucinated content. This work represents a step forward in mitigating such cases by providing the necessary content.

10 Ethics Statement

This paper advances the integration of Knowledge Graphs and Large Language Models to enhance their capability for domain-specific question answering. This work can facilitate the development of chatbots that provide responses limited to certain answers, aiding in meeting industry-standard protocols and ensuring adherence to privacy guidelines.

By anchoring the model's reasoning processes in KGs, we aim to increase the transparency of LLM outputs. Each reasoning step is grounded in structured, retrievable information, enabling users to trace the origin of a given response. We admit that fully interpretable AI remains a challenge, and transparency remains a primary focus for better relying on such complex models.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690. 594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

- Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *ArXiv preprint*, abs/2310.05845.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783.
- Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. ArXiv preprint, abs/2312.10997.
- Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, pages 132876– 132907. Curran Associates, Inc.
- Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *ArXiv preprint*, abs/2404.07103.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.
- Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-searchpredict: Composing retrieval and language models for knowledge-intensive nlp. *ArXiv preprint*, abs/2212.14024.
- Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421, Singapore. Association for Computational Linguistics.
- Miyoung Ko, Sue Hyun Park, Joonsuk Park, and Minjoon Seo. 2024. Investigating how large language models leverage internal knowledge to perform complex reasoning. *ArXiv preprint*, abs/2406.19502.

758

759

760

761

762

- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles.
- Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.
- Feiyang Li, Peng Fang, Zhan Shi, Arijit Khan, Fang Wang, Dan Feng, Weihao Wang, Xin Zhang, and Yongjian Cui. 2025. Cot-rag: Integrating chain of thought and retrieval-augmented generation to enhance reasoning in large language models. *arXiv* preprint arXiv:2504.13534.

664

675

682

693

698

701

704

- Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.
 - Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2024. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. *ArXiv preprint*, abs/2410.08815.
 - Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Haoran Luo, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, et al. 2023. Chatkbqa: A generatethen-retrieve framework for knowledge base question answering with fine-tuned large language models. *ArXiv preprint*, abs/2310.08975.
- Haoyan Luo and Lucia Specia. 2024. From understanding to utilization: A survey on explainability for large language models. *ArXiv preprint*, abs/2401.12874.
- Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations.*
- Elan Markowitz, Anil Ramakrishna, Jwala Dhamala, Ninareh Mehrabi, Charith Peris, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. 2024. Tree-oftraversals: A zero-shot reasoning algorithm for augmenting black-box language models with knowledge

graphs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Vol-ume 1: Long Papers)*, pages 12302–12319, Bangkok, Thailand. Association for Computational Linguistics.

- Costas Mavromatis and George Karypis. 2024. Gnnrag: Graph neural retrieval for large language model reasoning. *ArXiv preprint*, abs/2405.20139.
- Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022. SKILL: Structured knowledge infusion for large language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1581–1588, Seattle, United States. Association for Computational Linguistics.
- Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *ArXiv preprint*, abs/2408.08921.
- Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *ArXiv preprint*, abs/2204.00498.
- Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.
- SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *ArXiv preprint*, abs/2401.01313.
- Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. 2022. Will we run out of data? limits of llm scaling based on human-generated data. *ArXiv preprint*, abs/2211.04325.
- Xinyi Wang, Alfonso Amayuelas, Kexun Zhang, Liangming Pan, Wenhu Chen, and William Yang Wang. 2024. Understanding reasoning ability of language models from the perspective of reasoning paths aggregation. In *Forty-first International Conference on Machine Learning*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. MindMap: Knowledge graph prompting sparks graph of thoughts in large language models. In *Proceedings* of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10370–10388, Bangkok, Thailand. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

763

764 765

766

767

768

769

770

772

774 775

776

777

778

779 780

781

782

783 784

785

786 787

788

790

- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *CoRR*, abs/2210.03629.
- Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *ArXiv preprint*, abs/2210.00063.
- Danna Zheng, Mirella Lapata, and Jeff Z Pan. 2024. Large language models as reliable knowledge bases? *ArXiv preprint*, abs/2407.13578.
- Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W Huang, Jie Fu, Xiang Yue, and Wenhu Chen. 2024. Structlm: Towards building generalist models for structured knowledge grounding. *ArXiv preprint*, abs/2402.16671.
 - Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: a dataset for llm question answering with external tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23.

A GRBench Statistics

792

799

801

802

Detailed statistics of the graphs in GRBench (Jin et al., 2024) are shown in Table 3. Academic Graphs contain 3 types of nodes: paper, author, venue. Literature Graphs contain 4 types of nodes: book, author, publisher and series. Healthcare Graph contains 11 types of nodes: anatomy, biological process, cellular component, compound, disease, gene, molecular function, pathway, pharmacologic class, side effect, and symptom. Questions are created according to multiple templates labeled as easy, medium, and hard, depending on the number of nodes required to give the answer.

Domain	Tonio	Graph S	Statistics	Data		
Domain	Topic	# Nodes	# Edges	# Templates	# Questions	
	Biology	$\sim 4M$	$\sim 39M$	14	140	
	Chemistry	$\sim 4M$	$\sim 30 M$	14	140	
Academic	Material Science	$\sim 3M$	$\sim 22M$	14	140	
	Medicine	$\sim 6M$	$\sim 30 M$	14	140	
	Physics	$\sim 2M$	$\sim 33M$	14	140	
Literature	Goodreads	$\sim 3M$	$\sim 22M$	24	240	
Healthcare	Disease	$\sim 47 K$	$\sim 4M$	27	270	
SUM	-	-	-	121	1210	

Table 3: Detailed	l statistics o	f the GRBench	(Jin et al.,	2024).
-------------------	----------------	---------------	--------------	--------

B LLM <-> KG Interaction Pipelines

Description of the two LLM + KG Interaction Pipelines in their CoT form:

- 1. Agent Figure 9
- 2. Automatic Graph Exploration Figure 10



Figure 9: Agent Pipeline: (1) Input Query, (2) Thought Generation (3) Action Selection, (4) Environment Observation from the Knowledge Graph. The process is repeated until termination action is generated or limit reached.



Figure 10: Automatic Graph Exploration Pipeline: (1) Input Query, (2) Thought Generation, (3) Entity Extraction (from query or thought with LLM), (4) Automatic Graph Search as described in Algorithm 1 (5) Query LLM for answer or continue

Algorithm 1 presents the pseudocode for the Automatic Graph Exploration described in Section 4.2.2.

Algorithm 1 Graph Exploration Algorithm
1: procedure GRAPHEXPLORE(<i>LLM</i> , <i>seen_entities</i> , <i>search_depth</i>)
2: $relevant_attributes, found_triples \leftarrow 0$
3: for depth in search_depth do
4: for <i>entity</i> in <i>seen_entities</i> do
5: if seen_entities[entity_id].visited == True then
6: Continue
7: else
8: $seen_entities[entity] \leftarrow Visited$
9: end if
10: $head_entity_name, entity_attributes, neighbors \leftarrow Graph[entity]$
11: $pruned_neighbors \leftarrow prune_relations(LLM, neighbors)$
12: $pruned_neighbors \leftarrow prune_entities(LLM, pruned_neighbors)$
13: $found_triples \leftarrow generate_triples(entity, pruned_neighbros)$
14: end for
15: $seen_entities \leftarrow Update(seen_entities, neighbors)$
16: if End?(<i>LLM</i> , <i>found_triples</i> , <i>relevant_attributes</i>) == True then
17: break
18: end if
19: end for
20: return <i>found_triples</i> , <i>relevant_attributes</i> , <i>seen_entities</i>
21: end procedure

C Performance results in plots

Figures 11 and 12 illustrate the performance results using the Rouge-L and GPT4score metrics, respectively, for the healthcare graph for all methods. The results were run on the LLama 3.1 Instruct models (8B, 70B, and 405B-FP8) and demonstrate the improved performance achieved through more complex reasoning and search strategies during inference.



Figure 11: Performance results using the Rouge-L metric on the healthcare graph of GRBench (Jin et al., 2024), comparing all methods with LLama 3.1 Instruct models of various sizes (8B, 70B, 405B-FP8). Experimental details are included in Section 5.

804 805

806 807



Figure 12: Performance results using the GPT4Score metric on the healthcare graph of GRBench (Jin et al., 2024), comparing all methods with LLama 3.1 Instruct models of various sizes (8B, 70B, 405B-FP8). Experimental details are included in Section 5.

D Results on GPT4Score

In this section, we present the results of the experiments described in Section 5 for all methods, using the GPT4Score metric. This metric calculates the percentage of "correct" answers as judged by GPT-4 when presented with both the correct and the generated answer. The tables in this section present the same data as in Tables 1 and 2, but evaluated using GPT4Score.

	Method		Model	Healthcare	Goodreads	Biology	Chemistry	Materials Science	Medicine	Physics
	Base		Llama 3.1 8B-Ins	11.85	13.33	10.71	11.43	7.86	7.87	9.29
			Llama 3.1 70B-Ins	12.96	19.17	10.00	12.14	11.43	11.43	12.86
			Llama 3.1 405B-Ins	15.55	26.67	12.86	12.14	12.14	13.57	12.14
nes			Llama 3.1 8B-Ins	11.85	21.67	12.86	10.00	10.00	8.57	7.86
seli	Text-RAC	ì	Llama 3.1 70B-Ins	12.22	27.5	12.14	13.57	13.57	13.57	12.86
Ba			Llama 3.1 405B-Ins	12.96	26.67	15.00	13.57	12.86	14.29	13.57
			Llama 3.1 8B-Ins	14.81	32.50	29.29	29.28	27.86	25.71	29.29
	Graph-RA	G	Llama 3.1 70B-Ins	17.04	32.92	39.29	40.71	43.57	34.29	40.00
			Llama 3.1 405B-Ins	18.15	31.67	37.14	42.86	40.00	36.43	41.43
			Llama 3.1 8B-Ins	18.15	32.5	20.71	19.28	25.00	14.29	21.43
	Agent		Llama 3.1 70B-Ins	32.59	43.75	50.00	51.43	50.00	48.57	46.43
hd To			Llama 3.1 405B-Ins	28.89	48.33	38.57	38.57	47.86	56.43	34.29
ΰĞ	Graph Explore		Llama 3.1 8B-Ins	22.22	36.67	35.00	30.71	29.29	29.29	32.86
			Llama 3.1 70B-Ins	27.78	32.92	45.71	49.29	40.00	40.00	44.29
			Llama 3.1 405B-Ins	28.89	48.33	38.57	38.57	47.86	56.43	34.29
			Llama 3.1 8B-Ins	30.49	55.14	43.33	41.67	44.05	36.43	39.52
		Score	Llama 3.1 70B-Ins	30.49	54.48	65.48	62.14	55.95	63.57	56.19
	Acout		Llama 3.1 405B-Ins	45.55	56.53	71.67	65.71	52.62	68.81	44.76
	Agent		Llama 3.1 8B-Ins	30.00	54.17	40.71	37.14	40.00	32.86	36.43
		Select	Llama 3.1 70B-Ins	39.63	56.67	65.00	67.14	62.86	60.71	55.55
hd T			Llama 3.1 405B-Ins	44.07	58.75	71.43	69.29	65.00	68.81	60.00
ËĔ			Llama 3.1 8B-Ins	21.48	41.10	32.86	31.67	31.43	32.14	35.24
		Score	Llama 3.1 70B-Ins	24.94	40.97	52.38	57.86	49.29	54.29	47.86
	Graph		Llama 3.1 405B-Ins	30.86	48.33	47.86	57.14	50.71	56.67	47.14
	Explore		Llama 3.1 8B-Ins	21.85	41.67	32.86	31.67	31.43	32.14	35.24
		Select	Llama 3.1 70B-Ins	30.37	42.08	54.29	57.14	47.86	52.14	46.43
			Llama 3.1 405B-Ins	31.48	48.75	45.00	57.86	48.86	57.14	45.71

Table 4: GPT4Score performance results on GRBench (Jin et al., 2024), comparing standard LLMs, Text-RAG, Graph-RAG, Graph-CoT, and Graph-ToT. Experiments are described in Section 5, using LLama 3.1 - Instruct backbone models with sizes 8B, 70B, and 405B.

810 811

809

Method		Model	Healthcare	Biology
	e	Llama 3.1 8B-Ins	29.88	32.86
	õ	Llama 3.1 70B-Ins	29.51	61.69
ent	S	Llama 3.1 405B-Ins	41.81	48.33
Ag.		Llama 3.1 8B-Ins	30.00	40.71
	ele	Llama 3.1 70B-Ins	39.63	69.83
	S	Llama 3.1 405B-Ins	44.81	72.86
e	e	Llama 3.1 8B-Ins	22.72	21.19
lor	C01	Llama 3.1 70B-Ins	24.20	48.57
Exp	S	Llama 3.1 405B-Ins	32.22	41.67
I qc		Llama 3.1 8B-Ins	22.59	19.28
ìral	ele	Llama 3.1 70B-Ins	32.96	52.86
0	Ś	Llama 3.1 405B-Ins	31.48	57.86

Figure 13: Graph-GoT results (GPT4Score) on GR-Bench with Llama 3.1 Instruct sizes 8B, 70B, and 405B.



Figure 14: Results decomposed into easy/medium/hard questions according to GPT4Score.

E Computational Analysis

	Method	Key Parame- ters	Approx. # LLM Calls	Approx. # KG Operations	Primary Growth Driver(s)
oT	Agent	n	$\mathcal{O}(n)$	$\mathcal{O}(n)$	n (linear)
Ŭ	Graph Explore	n, d	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot \operatorname{Cost}_{\operatorname{Explore}}(d))$	n, d
ToT	Agent	D_{\max}, k, t	$\mathcal{O}\left(k \cdot rac{t^{D_{\max}}-1}{t-1} ight)$	Same as LLM Calls	D_{\max}, k, t (exponential in D_{\max})
	Graph Explore	D_{\max}, k, t, d	$\mathcal{O}\left(k \cdot rac{t^{D_{\max}}-1}{t-1} ight)$	$\mathcal{O}\left(k \cdot rac{t^{D_{\max}} - 1}{t - 1} \cdot \operatorname{Cost}_{\operatorname{Explore}}(d) ight)$	D_{\max}, k, t, d
GoT	Agent	D_{\max}, k, t	$\mathcal{O}\left(k \cdot \frac{t^{D_{\max}} - 1}{t - 1} + \sum_{i=1}^{D_{\max}} \right)$	Same as LLM Calls $\left \frac{k \cdot t^i}{2} \right $	D_{\max} , k, t (aggregation tion adds extra cost)
	Graph Explore	D_{\max}, k, t, d	Same as Agent	$\mathcal{O}\left(\text{LLM Calls} \cdot \text{Cost}_{\text{Explore}}(d)\right)$	D_{\max}, k, t, d

Table 5: Theoretical computational complexity comparison of reasoning methods. Parameters: n (reasoning steps), D_{max} (tree depth), k (branching factor), t (paths retained), d (KG search depth), and $\text{Cost}_{\text{Explore}}(d)$ (cost per KG search). GoT includes pairwise aggregation of thoughts at each depth.

The computational analysis summarized in Table 5 highlights the trade-offs between reasoning power and computational cost when grounding LLMs with Knowledge Graphs (KGs). The *Agent*-based methods scale linearly with the number of reasoning steps or tree nodes, with CoT representing the lowest cost baseline. In contrast, *Automatic Graph Exploration* methods introduce additional overhead via entity extraction, multi-hop traversal up to a fixed max_depth, and LLM-based pruning at each step.

Among reasoning strategies, ToT introduces exponential growth in cost with respect to depth due to its exploration of k branches and selection of t continuations per level. GoT further amplifies this by incorporating aggregation transformations that attempt to merge every pair of thoughts at each depth, leading to an additional cost proportional to $\sum_{i=1}^{D_{\text{max}}} \left| \frac{k \cdot t^i}{2} \right|.$

Importantly, our experiments reveal that the higher com-

plexity of GoT does not consistently translate to improved accuracy compared to ToT, suggesting diminishing returns. While the model size (e.g., 8B, 70B, 405B) influences the latency and memory footprint of each LLM call, it does not affect the algorithmic complexity classes shown. Hence, selecting a strategy



Figure 15: Comparison of computational costs across different reasoning strategies — CoT, ToT, GoT — as a function of reasoning steps. GoT exhibits the highest cost due to merge operations and graph traversal.

834

requires balancing reasoning depth with feasible compute budgets.

F Prompts Archive

In this section, we gather the necessary prompts essential for implementing the proposed methodologies.

Agent
Agent Step: Solve a question answering task with interleaving Thought, Interaction with Graph, Feedback from Graph steps. In Thought step, you can think about what further information is needed, and In Interaction step, you
can get feedback from graphs with four functions:(1) RetrieveNode[keyword], which retrieves the related node from the graph
<pre>(2) NodeFeature[Node, feature], which returns the detailed attribute information of Node regarding the given "feature" key. (3) NodeDegree[Node, neighbor_type], which calculates the number of "neighbor_type" neighbors of the node Node in the graph</pre>
(4) NeighbourCheck[Node, neighbor_type], which lists the "neighbor_type" neighbours of the node Node in the graph and returns them.You may take as many steps as necessary.Here are some examples:
{examples}
Please answer by providing node main feature (e.g., names) rather than node IDs.
<pre>Generate the next step. Definition of the graph: {graph_definition} Question: {question} {scratchpad}</pre>

Automatic Graph Exploration

Search Thought: Given the previous thoughts, generate the next thought to answer the provided question.

Your end goal is to answer the question step by step. For context, you are also provided with some knowledge triples from a knowledge base. Follow the format of the examples to generate the next thought.

{examples}

Graph Definition: {graph_definition}
Question: {question}
Knowledge Triples:
{triples}
Previous thoughts:
{thoughts}
Related Entity Attributes:
{attributes}
Next Thought:

Search End?: Your are provided with the an original question, the associated subquestion thoughts and their corresponding knowledge graph

⁸³⁷

triples (head_entity -> relation -> tail_entity). Your task is to answer whether it's sufficient for you to answer the original question (Yes or No). You are provided with examples. You should follow the same format as in the examples, writing 'Yes' or 'No' within brackets at the beginning of the answer. (Examples) Task: Question: {question} Thoughts: {thoughts} Knowledge Triples: {triples} Entity Attributes: {attributes} Answer:

Entity Extraction: Given the provided text, extract the relevant entities that
may appear in a knowledge base. Return the answer at the end with brackets
relevant entities as shown in the following examples. If there are several
entities, separate them with commas.
(Examples)
Task: Text: {text}

Relevant Entities:

Prune Relations: From the given entity and relations, select only the relevant relations to answer the question. Provide the answer at the end with bracketsanswer , as shown in the following example. (Examples) Question: {question} Head Entity: {entity} Relations: {relations} Answer:

Prune entities: You are provided with a question, a head entity, a relation and tail entity or entities from a knowledge base. Select the tail entity or entities to answer the question. Return the tail entity or entities at the end with brackets relevant entity or entities, as shown in the following examples.

(Examples)
Question: {question}
Head Entity: {head_entity}
Relation: {relation}
Tail Entities: {tail_entities}
Relevant Entities:

Search Attributes: Is any of the attributes relevant to answer the question?
Return the answer at the end with brackets answer , as shown in the following
examples.
(Examples)
Question: {question}
Entity: {entity}
Attributes: {attributes}
Relevant Attributes:

State Evaluators

Selection Vote: Given a question, you need to select the possible chain of thought that may lead to the correct answer with higher probablity. You are provided with several choices with thouhgts and related triples from a knowledge base. Decide which choice is most promising to complete the task. Analyze each choice in detail, then conclude in the last line: "The best choice is s", where s the integer id of the choice.

(*Examples*) Question: {*question*} Choices: {*choices*} (+ attributes/triples) Answer:

Score Vote: Generate a score for the given reasoning chain. The score represents the probability that the chain will lead to the correct answer. The chains contain interleaved thoughts and related triples from a knowledge base. Some chains may not be complete, but you need to judge the steps that are provided. The score can be any floating number between 0 and 1. (*Examples*)

Question: {*question*} Thought Chain: {*thoughts*} (+ attributes/triples) Score:

Graph of Thoughts

Graph of Thought Merge: Generate the next thought for the merged chain of thoughts. You are provided with the question, two chains of thoughts, and the corresponding merged chain of thought. Identify inconsistencies or errors from the previous chains and provide the next thought for the merged chain. You should follow the same format as in the examples.

(*Examples*) Question: {*question*} Chain 1: {*chain_1*} (+triples/attributes) Chain 2: {*chain_2*} (+triples/attributes) Merged Chain: {*merged_chain*}

Next Thought:

G Examples

Example 1: Multi-step Reasoning over Graph

Question: What anatomy can be expressed by gene KRT39?

Ground Truth Answer: *head, skin of body* **Model Answer:** *head, skin of body*

Thought 1: The question is related to a gene node (KRT39). We need to find this node in the graph. Action 1: RetrieveNode[KRT39] Observation 1: The ID of the node is 390792.

Thought 2: We need to check the 'Anatomy-expresses-Gene' neighbors of this gene node.

Action 2: NeighbourCheck[390792, Anatomy-expresses-Gene]

Observation 2: The neighbors are ['UBERON:0000033', 'UBERON:0002097'].

Thought 3: Retrieve names of the anatomy nodes. **Action 3:** NodeFeature[UBERON:0000033, name], NodeFeature[UBERON:0002097, name] **Observation 3:** UBERON:0000033 \rightarrow head, UBERON:0002097 \rightarrow skin of body.

Thought 4: These are the anatomy terms expressed by the gene.

Action 4: Finish[head, skin of body]

Table 6: Comparison of two LLM-generated reasoning traces for the same question using different strategies: multi-step graph traversal (Example 1) vs. direct triple lookup (Example 2).

Example 2: Direct Triple Lookup

Question: What anatomy can be expressed by gene KRT39? Ground Truth Answer: head, skin of body Model Answer: head, skin of body

Thought 1: KRT39 is a gene that is known to be expressed in two anatomical regions. **Found Triples:**

- "KRT39" -> Anatomy-expresses-Gene -> head
- "KRT39" -> Anatomy-expresses-Gene -> skin of body

Action: Finish[head, skin of body]