

Divide by Question, Conquer by Agent: SPLIT-RAG with Question-Driven Graph Partitioning

Ruiyi Yang¹ Hao Xue^{2,1} Imran Razzak^{3,1}
Shirui Pan⁴ Hakim Hacid⁵ Flora D. Salim¹

¹UNSW Sydney, Sydney, NSW, Australia

²The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China

³Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE

⁴Griffith University, Queensland, Australia

⁵Technology Innovation Institute, Abu Dhabi, UAE

{ruiyi.yang, flora.salim}@unsw.edu.au; haoxue@hkust-gz.edu.cn

imran.razzak@mbzuai.ac.ae; s.pan@griffith.edu.au; hakim.hacid@tii.ae

Abstract

Retrieval-augmented generation over knowledge graphs (KGs) can improve the factual grounding of large language models, but efficient retrieval remains challenging when the underlying graph is large and multi-hop evidence is distributed across different regions. We present **SPLIT-RAG**, an efficiency-oriented multi-agent KG-RAG framework that organises a KG into reusable, question-aligned partitions using reasoning patterns observed in training questions. At inference time, SPLIT-RAG forms a query-specific retrieval plan, activates only a small set of relevant partition-specialised agents, and aggregates their retrieved evidence before final answer generation. To improve reliability when evidence from different agents is combined, the framework applies compatibility-based filtering over retrieved triples prior to synthesis. Experiments on WebQSP, CWQ, and MetaQA show that SPLIT-RAG achieves strong performance against recent KG+LLM baselines while reducing end-to-end latency under matched budgets. Component ablations and model-tier analyses further show that selective multi-agent retrieval is central to the performance gain, and that lightweight retrieval agents paired with a stronger head agent provide an effective accuracy–latency trade-off. Code is available at <https://github.com/cruiseresearchgroup/SPLIT-RAG>.

1 Introduction

Large language models (LLMs) can produce unsupported or outdated answers when the required facts are absent from, or no longer reflected in, their parametric knowledge (Zhang et al., 2025; Kandpal et al., 2023). Retrieval-Augmented Generation (RAG) addresses this limitation by grounding generation in external evidence without requiring model retraining (Zhu et al., 2021; Gao et al.,

2023b; Zhao et al., 2024). Knowledge graphs (KGs) are particularly attractive retrieval sources because their entity–relation structure supports explicit multi-hop reasoning and structured evidence tracing.

However, scaling KG-based RAG to large graphs remains challenging. A complex question may require evidence spanning multiple relations, while a simple question should not require searching a large portion of the graph. Recent KG+LLM systems improve path or subgraph retrieval, but they still face an important accuracy–efficiency trade-off: expanding retrieval improves the chance of covering useful evidence, yet increases latency and introduces more irrelevant or weakly supported facts into the generation context. Moreover, when evidence is retrieved from multiple graph regions or agents, the final generator must combine partially overlapping or potentially incompatible triples in a controlled manner.

This work studies whether previously observed question-reasoning patterns can be reused to organise and route KG retrieval. The central intuition is that training questions reveal recurring retrieval structures: questions with related entity types and relation paths often depend on similar graph regions. Instead of retrieving over the complete KG for every query, a system can construct reusable question-aligned partitions offline and activate only the partitions most relevant to a new question at inference time.

Based on this intuition, we propose **SPLIT-RAG**, an efficiency-oriented multi-agent framework for knowledge-graph retrieval-augmented generation. SPLIT-RAG first constructs reusable KG partitions from reasoning-path fragments associated with training-question patterns. It then assigns partition groups to specialised retrieval agents. At inference time, a query-specific planner uses

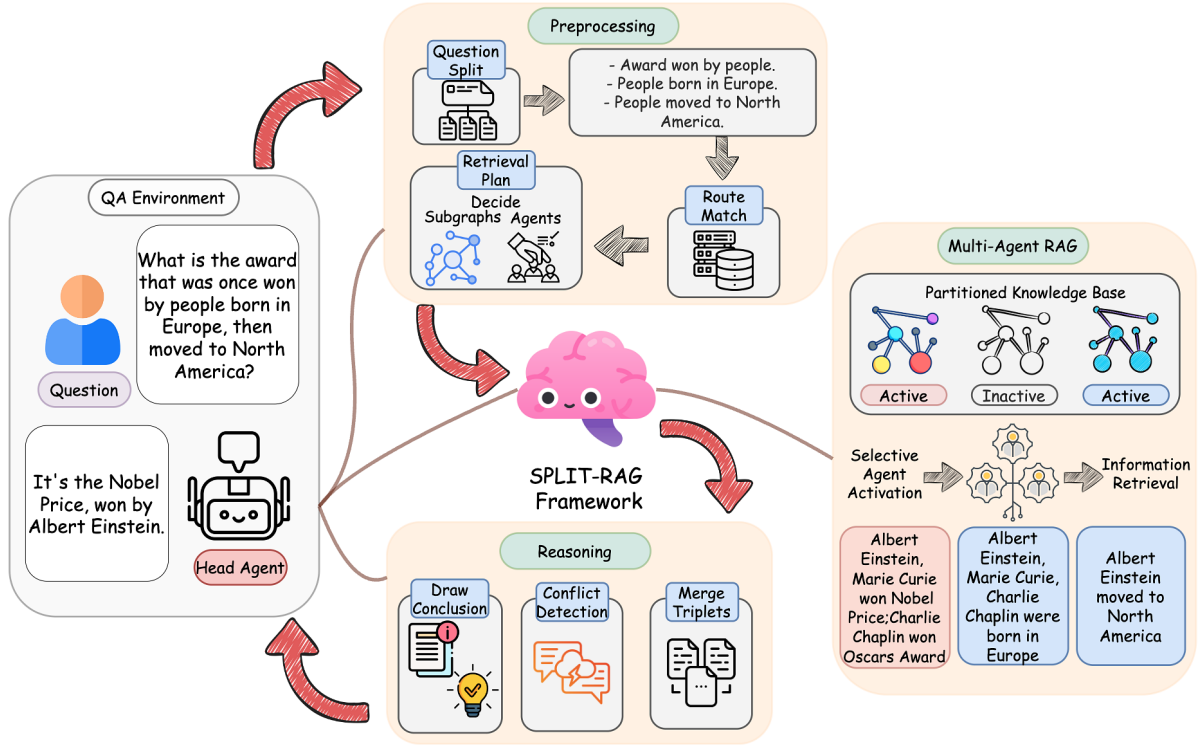


Figure 1: Illustration of SPLIT-RAG. Training-question reasoning patterns are used to construct question-aligned graph partitions and assign specialised retrieval agents. For a new query, the planner selects relevant partition-agent groups, retrieves supporting triples in parallel, and applies compatibility-based evidence filtering before head-agent answer generation.

entity-type similarity and, when available, a predicted relation-path sketch to activate a small set of relevant agents. The selected agents retrieve evidence only from their assigned partitions in parallel, after which a head agent generates the answer from a filtered set of compatible triples and textualised evidence.

This design differs from prior KG+LLM retrieval systems in three respects. First, SPLIT-RAG uses training-question patterns to construct reusable retrieval regions before inference, rather than deriving each retrieval space from the current query alone. Second, it couples these partitions with specialised agents and performs query-conditioned partition-agent routing, allowing retrieval effort to be allocated selectively. Third, it applies compatibility-based evidence filtering before final generation, providing a lightweight mechanism for refining evidence collected from multiple retrieval agents.

Our contributions are as follows:

- **Question-aligned KG partitioning.** We introduce an offline partitioning objective that organises a KG into compact, reusable regions aligned with recurring training-question rea-

soning patterns, while controlling partition size and fragmentation.

- **Partition-specialised multi-agent retrieval.** We develop a routing framework that selects relevant partition-agent groups for each incoming question and executes subgraph-specific retrieval in parallel, reducing unnecessary graph exploration.
- **Empirical analysis of effectiveness and efficiency.** We evaluate SPLIT-RAG on WebQSP, CWQ, and MetaQA against recent KG+LLM baselines, and analyse latency, component ablations, model-tier assignments, and incomplete-KG settings. The results show that selective multi-agent retrieval yields a strong accuracy-latency trade-off, with lightweight retrieval agents remaining effective when paired with a capable head agent.

2 Related Work

2.1 RAG and Graph-Based Retrieval

Retrieval-augmented generation (RAG) grounds LLM outputs in external evidence, improving factuality and supporting up-to-date or domain-specific knowledge access (Lewis et al., 2021; Jiang et al.,

Table 1: Comparison with representative KG+LLM systems. ✓ = present, × = absent, ○ = partial.

Method	Q-retrieval	Off. part.	MA spec.	Conflict	P-A route
RoG (Luo et al., 2023)	✓	×	×	×	×
ToG (Sun et al., 2023)	✓	×	×	×	×
StructGPT (Jiang et al., 2023a)	○	×	×	×	×
SubgraphRAG (Li et al., 2024a)	✓	×	×	×	×
Mindful-RAG (Agrawal et al., 2024)	✓	×	×	○	×
GcR (Luo et al., 2024)	✓	×	×	×	×
GNN-RAG (Mavromatis and Karypis, 2025)	✓	×	×	×	×
RPO-RAG (Um et al., 2026)	✓	×	×	×	×
SPLIT-RAG (ours)	✓	✓	✓	✓	✓

2023b; Zhao et al., 2024; Fan et al., 2024; Gao et al., 2023b; Yu et al., 2024; Pan et al., 2024). Recent graph-based RAG methods further exploit entity–relation structures to support interpretable multi-hop retrieval (Peng et al., 2024; Hu et al., 2024). Recent approaches further improve KGQA retrieval through efficient graph-neural retrieval (Mavromatis and Karypis, 2025), differentiable subgraph retrieval (Gao et al., 2025), or relation-aware preference optimisation for small LLMs (Um et al., 2026). However, retrieving over large graphs can still introduce substantial redundancy and latency. SPLIT-RAG addresses this issue by routing each question to a small set of relevant graph partitions rather than searching the full knowledge base.

2.2 Knowledge Graph Partitioning

Classical graph partitioning methods optimise structural objectives such as edge-cut or vertex-cut, including Kernighan–Lin, Fiduccia–Mattheyses, METIS, and JA-BE-JA (Buluç et al., 2016; Kim et al., 2011; Kernighan and Lin, 1970; Fiduccia and Mattheyses, 1988; Karypis and Kumar, 1997; Rahimian et al., 2015). Related KG-scale processing methods consider compression, alignment, landmark-based subgraph construction, or mapping-based partitioning to improve scalability (Qin et al., 2020; Trisedya et al., 2023; Arenas-Guerrero et al., 2024). In contrast, SPLIT-RAG partitions the KG according to reasoning paths observed in training questions, so that the resulting subgraphs are aligned with downstream QA retrieval patterns rather than graph structure alone.

2.3 LLM-Based Multi-Agent Systems

LLM-based multi-agent systems coordinate specialised agents across software development, engineering, embodied reasoning, simulation, recommendation, and financial decision-making tasks (Li et al., 2024b; Dong et al., 2024; Wu et al., 2024;

Mehta et al., 2023; Xia et al., 2023; Ahn et al., 2022; Huang et al., 2022; Gao et al., 2023a; Zhang et al., 2024; Guo, 2023; Zhao et al., 2023; He et al., 2024). SPLIT-RAG instantiates this idea for KGQA: agents are assigned to graph partitions, and only agents associated with a query-specific retrieval plan are activated at inference time.

2.4 Positioning of SPLIT-RAG

Table 1 positions SPLIT-RAG against representative KG+LLM systems along five design dimensions. Existing methods already support query-conditioned retrieval or graph-based reasoning, including subgraph and path retrieval. SPLIT-RAG differs in organising the KG into reusable, question-aligned partitions before inference, assigning specialised retrieval agents to these partitions, and routing each incoming question to a small set of relevant partition–agent groups. It further applies compatibility-based evidence filtering before final answer generation.

3 Divide by Question, Conquer by Agent: SPLIT-RAG Framework

SPLIT-RAG is an efficiency-oriented KG-RAG framework that routes each input question to a small set of specialised retrieval agents over question-aligned graph partitions. Let $D = (\mathcal{V}, \mathcal{E})$ be a knowledge graph and $\mathcal{Q}_{\text{train}}$ a set of training questions with recorded reasoning paths. An offline indexer τ constructs a partitioned knowledge base $\hat{\mathcal{K}} = \{\hat{D}_1, \dots, \hat{D}_M\}$ and assigns partitions to agents. At inference time, a planner ψ selects the relevant agents and subgraphs for a new question q , after which a head generator produces the final answer:

$$\hat{\mathcal{K}} = \tau(\mathcal{Q}_{\text{train}}, D), \quad F(q, D) = \hat{\mathcal{G}}(q, \psi(q, \hat{\mathcal{K}})) \quad (1)$$

The framework contains four stages: question-guided graph partitioning, agent allocation,

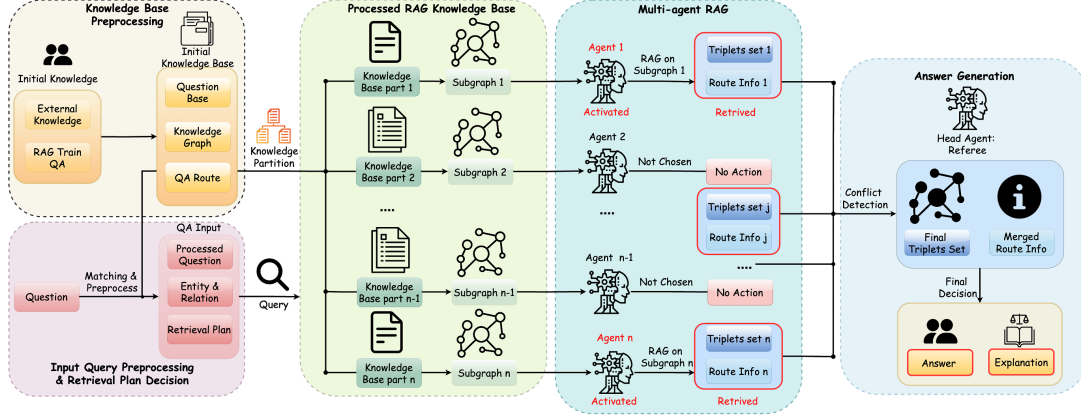


Figure 2: Overview of SPLIT-RAG. Training-question reasoning patterns guide graph partitioning and agent assignment. At inference time, a query-specific retrieval plan activates only relevant agents before compatibility-based evidence aggregation and answer generation.

query-time routing and parallel retrieval, and compatibility-based evidence aggregation.

3.1 Question-Guided Graph Partitioning

Question and path representations. Each training question q_i is represented by: (i) an entity-type view $\mathbf{Q}_{e,i}$, in which entity mentions are replaced by their KG types; and (ii) a reasoning path

$$\mathbf{P}_i = (e_1, r_1, e_2, \dots, r_{h_i}, e_{h_i+1})$$

that connects entities required for answering q_i . The entity-type views are clustered into question-pattern groups $c_i \in \mathcal{C}$. Each reasoning path is decomposed into consecutive at-most-two-hop fragments:

$$\phi(\mathbf{P}_i) = \{(e_j, r_j, e_{j+1}, \dots, e_{j+\ell}) : 1 \leq \ell \leq 2\}. \quad (2)$$

Partition objective. The goal is to construct compact graph regions that preserve frequently reused reasoning patterns. Let Z denote the partition assignment of a training-path fragment and let C denote the question-pattern cluster from which that fragment originates. A useful partition should make the activated graph region informative of the associated question pattern while avoiding excessively large or fragmented partitions. We therefore optimise

$$\mathcal{J}(\mathcal{S}) = I(C; Z) - \lambda \sum_{s \in \mathcal{S}} \left(\frac{|\mathcal{V}_s|}{|\mathcal{V}|} \right)^2 - \mu |\mathcal{S}|, \quad (3)$$

subject to

$$|\mathcal{V}_s| \leq \eta_{\max}, \quad \forall s \in \mathcal{S}, \quad \bigcup_{s \in \mathcal{S}} \mathcal{E}_s = \mathcal{E}. \quad (4)$$

The mutual-information term rewards partitions associated with coherent question patterns; the quadratic size penalty discourages a small number of oversized partitions; and the final term avoids unnecessary fragmentation.

In practice, path fragments initialise seed regions, which are greedily merged according to the marginal gain in Eq. (3). Triples not observed in training reasoning paths are assigned to the nearest compatible region using their entity and relation representations, ensuring that the resulting partition covers the full KG. Boundary entities may appear in multiple partitions when required to retain cross-partition reasoning paths. Algorithm 1 in Appendix D provides the partition-construction procedure.

3.2 Question-Centric Agent Allocation

Let $\hat{\mathcal{K}} = \{\hat{D}_1, \dots, \hat{D}_M\}$ be the graph partitions. For each training question q_i , define its partition coverage set as

$$B_{ij} = \mathbb{I}[\exists p \in \text{PATHS}(q_i) : p \cap \hat{D}_j \neq \emptyset]. \quad (5)$$

$$\mathcal{C}_i = \{\hat{D}_j \mid B_{ij} = 1\}. \quad (6)$$

Each agent a_ℓ is assigned a group of partitions $\mathcal{G}_\ell \subseteq \hat{\mathcal{K}}$. The allocation objective minimises the number of agents required by training questions:

$$\min_{\{\mathcal{G}_\ell\}_{\ell=1}^L} \mathcal{L}_{\text{coord}} = \sum_{i=1}^{|\mathcal{Q}_{\text{train}}|} |\{\ell : \mathcal{G}_\ell \cap \mathcal{C}_i \neq \emptyset\}|, \quad (7)$$

subject to

$$\bigcup_{\ell=1}^L \mathcal{G}_\ell = \hat{\mathcal{K}}, \quad |\mathcal{G}_\ell| \leq N_{\max}, \quad \text{Coh}(\mathcal{G}_\ell) \geq \theta_{\text{coh}} \quad (8)$$

Here, $\text{Coh}(\cdot)$ measures semantic similarity among partitions assigned to the same agent. The objective places partitions frequently used together under the same agent, reducing coordination overhead at inference time. We use a greedy coverage-co-occurrence allocation procedure, detailed in Appendix D, Algorithm 2.

3.3 Query-Time Routing and Parallel Retrieval

Given a new question q , the planner first forms its entity-type representation $\mathbf{Q}_{e,q}$. Optionally, it also predicts a relation-path sketch $\hat{\mathbf{p}}_q$ directly from the question. Importantly, $\hat{\mathbf{p}}_q$ is produced without access to gold test paths or answers; its generation cost is included in end-to-end latency.

For each training question q_i , routing similarity is computed as

$$\begin{aligned} s_{\text{sem}}(q, q_i) &= \cos(\text{Emb}(\mathbf{Q}_{e,q}), \text{Emb}(\mathbf{Q}_{e,i})), \\ s_{\text{path}}(q, q_i) &= \text{PathOverlap}(\hat{\mathbf{p}}_q, \mathbf{p}_i), \\ \text{Sim}(q, q_i) &= \gamma_s s_{\text{sem}}(q, q_i) + \gamma_p s_{\text{path}}(q, q_i). \end{aligned} \quad (9)$$

where $\gamma_p = 0$ when no path sketch is used. Let

$$\mathcal{N}_\theta(q) = \{q_i : \text{Sim}(q, q_i) \geq \theta_{\text{sim}}\}$$

be the retrieved training neighbours. Their associated agents form the similarity-based candidate set

$$\mathcal{A}_{\text{sim}}(q) = \bigcup_{q_i \in \mathcal{N}_\theta(q)} \mathcal{A}(q_i). \quad (10)$$

$$\mathcal{A}_{\text{path}}(q) = \left\{ a_\ell : \begin{array}{l} \exists \tilde{p} \in \phi(\hat{\mathbf{p}}_q), \exists \hat{D}_j \in \mathcal{G}_\ell, \\ \text{Compatible}(\tilde{p}, \hat{D}_j) = 1 \end{array} \right\}. \quad (11)$$

Here, $\text{Compatible}(\tilde{p}, \hat{D}_j)$ indicates that partition $\hat{D}_j$ contains entity–relation patterns compatible with the predicted path fragment $\tilde{p}$. When a predicted path sketch is available, partitions containing compatible relation fragments additionally produce $\mathcal{A}_{\text{path}}(q)$. Let $d(q) = 1$ indicate that a retrieved neighbour exceeds the direct-routing threshold θ_{direct} . The activated agents are

$$\mathcal{A}^*(q) = \begin{cases} \mathcal{A}_{\text{sim}}(q), & d(q) = 1, \\ \mathcal{A}_{\text{sim}}(q) \cup \mathcal{A}_{\text{path}}(q), & d(q) = 0. \end{cases} \quad (12)$$

Each selected agent receives a sub-task derived from the transferred question pattern or predicted relation fragments, and searches only within its assigned partitions. For an agent sub-task t and

candidate KG path p , we use

$$\text{Match}(p, t) = \frac{|\text{Ent}(p) \cap \text{Ent}(t)|}{\max(1, |\text{Ent}(t)|)}. \quad (13)$$

Paths whose matching score exceeds θ_{match} contribute structured triples $\mathcal{T}_i$ and textualised evidence $\mathcal{E}_i$. All selected agents operate in parallel. The implementation of relation-path sketch prediction, $\text{PathOverlap}(\cdot, \cdot)$, and $\text{Compatible}(\cdot, \cdot)$ is specified in Appendix F. Algorithms 3 and 4 in Appendix D provide the complete routing and parallel retrieval procedures.

3.4 Compatibility-Based Evidence Aggregation

Let $\mathcal{T}_{\text{all}} = \bigcup_{a_i \in \mathcal{A}^*(q)} \mathcal{T}_i$ and $\mathcal{E}_{\text{all}} = \bigcup_{a_i \in \mathcal{A}^*(q)} \mathcal{E}_i$ denote the collected triples and textual evidence. Each retrieved triple $\tau \in \mathcal{T}_{\text{all}}$ is assigned a support score:

$$w(\tau) = \alpha \max_{i: \tau \in \mathcal{T}_i} \text{Conf}(a_i) + \beta \text{Freq}(\tau) + \gamma \text{Entail}(\mathcal{E}_{\text{all}}, \tau) \quad (14)$$

where $\text{Conf}(a_i)$ is estimated from training-time retrieval success, $\text{Freq}(\tau)$ measures multi-agent support, and $\text{Entail}(\mathcal{E}_{\text{all}}, \tau)$ is produced by a lightweight evidence verifier.

To avoid incorrectly rejecting valid multi-valued KG relations, two triples are treated as conflicting only when they are explicitly contradictory or assign different objects to a relation identified as functional:

$$\text{Conflict}(\tau_1, \tau_2) = 1 \quad \text{iff} \quad \tau_1 \models \neg \tau_2 \vee \tau_2 \models \neg \tau_1 \vee \text{FuncCol}(\tau_1, \tau_2). \quad (15)$$

We construct a compatibility graph whose vertices are retrieved triples and whose edges connect non-conflicting pairs. Rather than claiming exact maximum-weight clique inference, we apply a greedy maximum-weight compatible-set heuristic, iteratively retaining the highest-scoring triple compatible with the current evidence set. The resulting set $\mathcal{T}_{\text{clean}}$ is supplied to the head agent:

$$\text{Answer}(q) = \text{LLM}_{\text{head}}(q, \mathcal{T}_{\text{clean}}, \mathcal{E}_{\text{all}}). \quad (16)$$

Implementation details for the support-score components, functional-relation identification, and greedy compatible-set construction are provided in Appendix H.

3.5 Conditional Evidence Preservation and Retrieval Cost

The benefit of SPLIT-RAG depends on whether routing activates partitions containing the evidence

required for a query. We therefore state two conditional properties rather than assuming perfect coverage for arbitrary unseen questions.

Proposition 1: Evidence preservation under routing coverage. For a question q , let $\mathcal{T}^*(q) \subseteq \mathcal{E}$ be a sufficient set of KG triples for deriving a correct answer, and let

$$\mathcal{U}(q) = \bigcup_{a_i \in \mathcal{A}^*(q)} \mathcal{G}_{a_i}$$

be the union of activated partitions. If

$$\mathcal{T}^*(q) \subseteq \mathcal{U}(q)$$

and the within-partition retriever returns all required triples present in $\mathcal{U}(q)$, then routing does not remove any supporting triple required for answering q relative to retrieval over the full graph.

Proposition 2: Retrieval-cost reduction under selective activation. Let $n = |\mathcal{E}|$ denote the total number of KG triples and let $n_q = |\mathcal{E}(\mathcal{U}(q))|$ denote the number of triples contained in the partitions activated for question q . For a retrieval operator with monotone search cost $C(\cdot)$, SPLIT-RAG has online cost

$$T_{\text{split}}(q) = T_{\text{plan}}(q) + C(n_q) + T_{\text{agg}}(q), \quad T_{\text{full}}(q) = C(n). \quad (17)$$

Therefore, SPLIT-RAG reduces online retrieval cost whenever

$$T_{\text{plan}}(q) + C(n_q) + T_{\text{agg}}(q) < C(n).$$

Under a linear candidate-scanning model, this becomes

$$T_{\text{split}}(q) = T_{\text{plan}}(q) + O(n_q), \quad T_{\text{full}}(q) = O(n),$$

so the efficiency gain is governed by the activated-graph ratio n_q/n and the planner overhead. We empirically measure both evidence coverage and end-to-end latency in Section 4. Formal proofs are provided in Appendix E.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate SPLIT-RAG on four KGQA benchmarks: WebQuestionsSP (WebQSP) (Yih et al., 2016), Complex WebQuestions (CWQ) (Talmor and Berant, 2018), and the 2-hop and 3-hop subsets of MetaQA (Zhang et al., 2018).

Table 2: Statistics of the KGQA benchmarks used in our experiments.

Dataset	#Train	#Test	Max Hop
WebQSP	2,826	1,628	2
CWQ	27,639	3,531	4
MetaQA-2Hop	119,986	114,196	2
MetaQA-3Hop	17,482	14,274	3

WebQSP and CWQ are constructed over Freebase and test reasoning over a large-scale KG, while MetaQA evaluates controlled multi-hop reasoning over a movie-domain ontology. We omit MetaQA-1Hop as it does not require multi-step retrieval. Dataset statistics are reported in Table 2.

Baselines. We compare against three groups of methods: (i) embedding-based KGQA models, including KV-Mem (Miller et al., 2016), GraftNet (Sun et al., 2018), PullNet (Sun et al., 2019), EmbedKGQA (Saxena et al., 2020), and TransferNet (Shi et al., 2021); (ii) LLM-only prompting without KG retrieval; and (iii) KG+LLM methods, including RoG (Luo et al., 2023), SubgraphRAG (Li et al., 2024a), ToG (Sun et al., 2023), GcR (Luo et al., 2024), GNN-RAG (Mavromatis and Karypis, 2025), RPO-RAG (Um et al., 2026) and a standard graph-based RAG baseline.

Metrics and implementation. We report Hit, Hits@1 (H@1), and F1. Hit measures whether any gold answer entity appears in the model output; H@1 measures whether the top predicted entity is correct; and F1 measures token-level overlap between predicted and gold answers.

Controlled comparison setting. KG+LLM baselines are reproduced based on unified Llama-3.1-8B head model, identical benchmark splits, and the same evaluation scripts. Latency comparisons further use matched retrieval and decoding budgets, as detailed in Appendix I, Table 13 and 14.

4.2 Main Results

Table 3 reports the overall KGQA performance. SPLIT-RAG achieves the strongest results among KG+LLM methods across all four benchmarks, including recent SOTA baselines such as GNN-RAG and RPO-RAG. The improvement is particularly relevant on WebQSP and CWQ, where the underlying Freebase graph is substantially larger and more relation-rich than MetaQA, supporting the effectiveness of question-conditioned partition routing for large-scale KG retrieval. On MetaQA,

Table 3: Performance comparison on KGQA benchmarks. Within the KG+LLM category, the **best** and second-best results are highlighted. Embedding-based and LLM-only results are included as reference comparisons where the corresponding metrics are available. KG+LLM baselines are reproduced based on unified Llama-3.1-8B head.

Overall Results										
Method	WebQSP			CWQ			MetaQA-2Hop		MetaQA-3Hop	
	Hit	H@1	F1	Hit	H@1	F1	Hit	H@1	Hit	H@1
Embedding methods										
KV-Mem (Miller et al., 2016)	N/A	46.7	N/A	N/A	21.1	N/A	N/A	82.7	N/A	48.9
GraftNet (Sun et al., 2018)	N/A	66.4	N/A	N/A	32.8	N/A	N/A	94.8	N/A	77.1
PullNet (Sun et al., 2019)	N/A	68.1	N/A	N/A	47.2	N/A	N/A	<u>99.9</u>	N/A	91.4
EmbedKGQA (Saxena et al., 2020)	N/A	66.6	N/A	N/A	45.9	N/A	N/A	98.8	N/A	94.8
TransferNet (Shi et al., 2021)	N/A	71.4	N/A	N/A	48.6	N/A	N/A	100	N/A	100
LLM methods (no KG retrieval)										
Llama3-8b	59.2	N/A	N/A	33.1	N/A	N/A	31.7	N/A	28.9	N/A
Davinci-003	48.3	N/A	N/A	35.6	N/A	N/A	42.5	N/A	25.3	N/A
ChatGPT	66.8	N/A	N/A	39.9	N/A	N/A	43.2	N/A	31.0	N/A
Gemini 2.0 Flash	65.3	N/A	N/A	41.1	N/A	N/A	46.9	N/A	32.2	N/A
Falcon-H1-7B-Instruct	67.4	N/A	N/A	45.8	N/A	N/A	48.1	N/A	33.5	N/A
KG+LLM methods										
Graph-RAG	77.2	73.1	67.7	58.8	54.6	53.9	88.1	85.4	81.3	78.2
RoG (Luo et al., 2023)	85.7	80.0	70.8	62.6	57.8	56.2	93.8	90.1	87.9	84.8
SubgraphRAG (Li et al., 2024a)	86.9	81.8	71.5	65.4	61.3	57.8	95.2	92.3	89.3	86.1
ToG (Sun et al., 2023)	82.1	78.5	69.8	63.3	59.8	56.4	91.6	88.9	87.3	84.7
GcR (Luo et al., 2024)	84.5	79.7	68.9	61.2	57.4	55.1	93.5	91.3	89.7	86.4
GNN-RAG (Mavromatis and Karypis, 2025)	<u>90.4</u>	85.3	73.5	69.5	65.1	62.2	<u>96.8</u>	<u>93.0</u>	91.5	85.9
RPO-RAG (Um et al., 2026)	89.1	<u>87.0</u>	<u>81.5</u>	<u>72.6</u>	<u>68.4</u>	<u>65.1</u>	95.7	92.2	93.4	<u>87.9</u>
SPLIT-RAG	92.8	89.2	82.2	73.3	69.1	66.2	97.1	95.4	93.4	88.1

embedding-based methods remain highly competitive due to the small and regular ontology; these datasets are used primarily to analyze controlled multi-hop behaviour and agent configurations.

Table 4: End-to-end latency per QA (ms) on WebQSP/CWQ under the unified budgets in Table 13.

Method	WebQSP Avg Time (ms)	CWQ Avg Time (ms)
Think-on-Graph (ToG)	41586.2	47623.0
Reasoning on Graphs (RoG)	34735.4	41344.2
GNN-RAG	30542.9	35239.1
SPLIT-RAG	29429.8	33726.8

Table 5: KG and Subgraph Size

Dataset	#Entity	#Relations	#Triplets	#Avg $\mathcal{G}$ Entity	$\mathcal{G}$ Coverage
WebQSP	2,566,291	7,058	8,309,195	68421.9	91.6
CWQ	2,566,291	7,058	8,309,195	40705.8	74.2
MetaQA	43,234	9	133,582	8651.2	99.9

As shown in Table 4, SPLIT-RAG achieves the lowest end-to-end latency on both WebQSP and CWQ under matched budgets. Importantly, it remains faster than the recent efficiency-oriented GNN-RAG baseline, while also obtaining stronger QA performance in Table 3. This result indicates that routing questions to a small set of specialised partitions improves the accuracy-latency trade-off rather than merely reducing retrieval cost.

Table 5 reports graph scale and routed-partition coverage. MetaQA reaches near-complete coverage because it contains a compact ontology

with regular reasoning patterns. On the substantially larger Freebase graph, SPLIT-RAG still achieves high coverage on WebQSP, while CWQ remains more challenging because its questions involve more diverse compositional relation patterns. These statistics help explain why CWQ provides the most demanding setting for routing generalisation.

4.3 Ablation Study

Table 6: Component ablation on WebQSP. A removes similarity-based planning; B replaces partition-specialised multi-agent retrieval with a single agent; C removes compatibility-based evidence filtering.

Method	Hit	H@1	F1	Avg.#Parts	Avg. Search Size
SPLIT-RAG	92.8	89.2	82.2	4.8	68,421.9
SPLIT-RAG - A	85.8	83.6	74.6	6.5	65,402.8
SPLIT-RAG - B	69.7	63.9	58.5	1	303,071.1
SPLIT-RAG - C	<u>87.6</u>	<u>85.1</u>	<u>79.1</u>	4.8	68,421.9

We conduct an ablation study on WebQSP to examine three components of SPLIT-RAG: (A) similarity-based retrieval planning using training-question patterns; (B) partition-specialised multi-agent retrieval; and (C) compatibility-based evidence filtering before head-agent generation. For variant A, we replace question-similarity-based planning with coarse subgraph-label matching. For variant B, the selected partitions are merged into a single retrieval region and searched by one agent.

Table 7: Full results on MetaQA-3Hop using different model tiers for agents (end-to-end per QA).

Group	$\mathcal{A}_i$	$\mathbf{A}_{\text{head}}$	Hit	H@1	Avg Time (s)
G1	Gemini 2.0 Flash	Gemini 2.0 Flash-Lite	82.1	80.3	28.1
G2	Gemini 2.5 Flash	Gemini 2.0 Flash-Lite	84.5	82.8	35.3
G3	Gemini 2.0 Flash-Lite	Gemini 2.0 Flash	91.5	88.3	19.2
G4	Gemini 2.0 Flash	Gemini 2.0 Flash	92.9	90.1	31.5
G5	Gemini 2.0 Flash-Lite	Gemini 2.5 Flash	93.7	91.0	27.6
G6	Gemini 2.0 Flash	Gemini 2.5 Flash	95.4	91.7	34.2

For variant C, all retrieved triples are supplied directly to the head agent without compatibility-based filtering.

Table 6 shows that all three components contribute to SPLIT-RAG. Removing similarity-based planning (A) reduces performance and increases the average number of activated partitions from 4.8 to 6.5, indicating that training-question patterns provide a more selective routing signal than coarse subgraph labels alone. Replacing partition-specialised multi-agent retrieval with a single agent (B) causes the largest degradation, while expanding the average searched region by more than four times. Moreover, the union of selected partitions is disconnected for 22.3% of questions in this variant, making it more difficult for a single retrieval agent to recover evidence distributed across routed graph regions. These results support the central design choice of searching question-relevant partitions through specialised retrieval agents rather than querying their merged union with one agent. Finally, removing compatibility-based evidence filtering (C) yields a smaller but consistent performance drop while leaving the retrieval space unchanged. Although explicit incompatibilities are detected in only 12.3% of evaluated questions, filtering them still improves answer performance, supporting its role as a complementary post-retrieval refinement step. Definitions of these diagnostic statistics are provided in Appendix G.1.

4.4 Accuracy–Latency Tradeoff & Model Flexibility: Is unifying all agents necessary?

We further investigate whether all agents require the same model capacity by varying the LLM tiers assigned to the subgraph agents $\{\mathcal{A}_i\}$ and the head agent $\mathbf{A}_{\text{head}}$. All configurations share the same planner, retriever, evidence caps, and decoding budgets; only the model tiers vary. The evaluated tiers

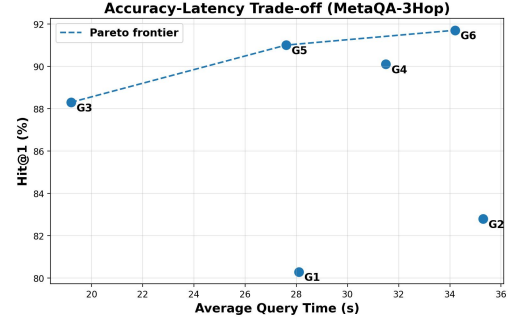


Figure 3: Accuracy–latency trade-off on MetaQA-3Hop measured by H@1. The Pareto frontier consists of G3, G5, and G6.

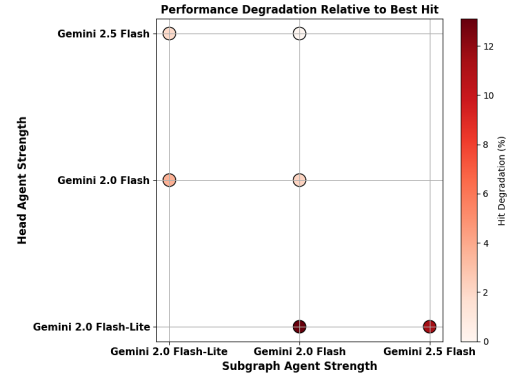


Figure 4: Performance degradation relative to the best H@1. Darker color indicates larger degradation.

are ordered by capacity as Gemini 2.0 Flash-Lite < Gemini 2.0 Flash < Gemini 2.5 Flash.

As shown in Table 7, configurations with a stronger head agent consistently provide a better accuracy–latency trade-off than those that allocate greater capacity only to subgraph agents. In particular, replacing Flash subgraph agents with Flash-Lite while retaining the strongest head model (G6 → G5) saves 6.6 seconds per question with only a small reduction in accuracy. These results suggest that SPLIT-RAG *does not require high-capacity models for every partition-specific retrieval call*. A controlled Llama-3.1 check confirms the same direction: 8B workers with an 8B head obtain 89.2 H@1 at 31.6 s; 3B workers with an 8B head ob-

Table 8: Robustness to incomplete KGs on WebQSP. Edges are dropped uniformly at random before partitioning; both forward and inverse copies of a dropped edge are removed together.

Setting	Hit	H@1	F1	Latency (s)
Full KG (0% drop, Llama-3.1-8B head)	92.8	89.2	82.2	29.4
10% dropped	84.5	83.3	76.1	28.8
30% dropped	72.3	69.2	65.5	25.3
50% dropped	59.2	57.1	55.9	19.7

tain 88.6 at 25.8 s; and 8B workers with a 3B head obtain 82.9 at 29.1 s.

4.5 Robustness under Incomplete Knowledge Graphs

Table 8 evaluates SPLIT-RAG when edges are removed before partition construction, thereby testing both partition formation and routed retrieval under incomplete evidence. The framework remains functional under moderate knowledge loss, while accuracy decreases monotonically as larger portions of answer-supporting paths are removed. Latency decreases accordingly because the searchable graph becomes smaller, rather than because the incomplete setting simplifies reasoning. This analysis characterises the dependence of SPLIT-RAG on the availability of supporting KG evidence.

5 Conclusion and Future Work

We presented **SPLIT-RAG**, an efficiency-oriented KG-RAG framework that constructs reusable question-aligned graph partitions, assigns specialised retrieval agents to them, and routes each query to a small set of relevant partition-agent groups. A compatibility-based filtering step refines multi-agent evidence before head-agent generation. Experiments on WebQSP, CWQ, and MetaQA demonstrate that SPLIT-RAG is competitive with recent KG+LLM retrieval systems while achieving lower end-to-end latency under matched budgets. Controlled ablations show that retaining partition-scoped retrieval states supplies the quality benefit, whereas parallel worker execution supplies the principal latency benefit. The model-tier analysis shows that lightweight retrieval workers can be effective when paired with a stronger head model. Additional incomplete-KG experiments characterise how retrieval quality changes when supporting graph evidence becomes unavailable. Future work will extend SPLIT-RAG to evolving and hybrid knowledge sources and develop stronger verification mechanisms for ambiguous retrieved evidence.

Limitations

SPLIT-RAG is designed for question answering over structured, triple-based knowledge graphs, where training questions provide useful signals for organising retrieval regions. Its current evaluation focuses on KGQA benchmarks with relatively stable schemas; extending the framework to hybrid corpora involving text, tables, or dynamically updated sources remains an important direction for future work.

Ethics Considerations

The experiments do not involve human participants, private user data, or the collection of personally identifiable information. SPLIT-RAG is intended as a research framework for improving the accuracy-efficiency trade-off of knowledge-grounded question answering. Although SPLIT-RAG includes compatibility-based evidence filtering, this mechanism should not be interpreted as a guarantee of factual correctness.

Acknowledgments

This research was partially supported by the Technology Innovation Institute (TII), Abu Dhabi, United Arab Emirates, which conducts cutting-edge applied research in artificial intelligence and advanced technologies. Also, this research used the NVIDIA H100 cluster provided by SHARON AI.

References

- Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Mindful-rag: A study of points of failure in retrieval augmented generation. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 607–611. IEEE.
- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang,

- Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2022. [Do as i can, not as i say: Grounding language in robotic affordances](#). *Preprint*, arXiv:2204.01691.
- Julián Arenas-Guerrero, David Chaves-Fraga, Jhon Toledo, María S Pérez, and Oscar Corcho. 2024. Morph-kgc: Scalable knowledge graph materialization with mapping partitions. *Semantic Web*, 15(1):1–20.
- Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2016. *Recent advances in graph partitioning*. Springer.
- Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. 2024. Self-collaboration code generation via chatgpt. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–38.
- Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.
- Charles M Fiduccia and Robert M Mattheyses. 1988. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*, pages 241–247.
- Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023a. S3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*.
- Guangze Gao, Zixuan Li, Chunfeng Yuan, Jiawei Li, Wu Jianzhuo, Yuehao Zhang, Xiaolong Jin, Bing Li, and Weiming Hu. 2025. D-rag: Differentiable retrieval-augmented generation for knowledge graph question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35386–35405.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.
- Fulin Guo. 2023. Gpt in game theory experiments. *arXiv preprint arXiv:2305.05516*.
- Junda He, Christoph Treude, and David Lo. 2024. Llm-based multi-agent systems for software engineering: Literature review, vision and the road ahead. *ACM Transactions on Software Engineering and Methodology*.
- Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*.
- Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. [Inner monologue: Embodied reasoning through planning with language models](#). *Preprint*, arXiv:2207.05608.
- Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.
- Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.
- Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR.
- George Karypis and Vipin Kumar. 1997. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.
- Brian W Kernighan and Shen Lin. 1970. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307.
- Jin Kim, Inwook Hwang, Yong-Hyuk Kim, and Byung-Ro Moon. 2011. [Genetic approaches for graph partitioning: a survey](#). In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO ’11, page 473–480, New York, NY, USA. Association for Computing Machinery.
- Patrick 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. 2021. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). *Preprint*, arXiv:2005.11401.
- Mufei Li, Siqi Miao, and Pan Li. 2024a. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *arXiv preprint arXiv:2410.20724*.
- Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024b. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinatearth*, 1(1):9.
- Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

- Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuanfang Li, Chen Gong, and Shirui Pan. 2024. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080*.
- Costas Mavromatis and George Karypis. 2025. Gnn-rag: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16682–16699.
- Nikhil Mehta, Milagro Teruel, Patricio Figueroa Sanz, Xin Deng, Ahmed Hassan Awadallah, and Julia Kiseleva. 2023. Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback. *arXiv preprint arXiv:2304.10750*.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.
- Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiaapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.
- 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 arXiv:2408.08921*.
- Kyle K Qin, Flora D Salim, Yongli Ren, Wei Shao, Mark Heimann, and Danai Koutra. 2020. G-crewe: Graph compression with embedding for network alignment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1255–1264.
- Fatemeh Rahimian, Amir H Payberah, Sarunas Girdzijauskas, Mark Jelasity, and Seif Haridi. 2015. A distributed algorithm for large-scale graph partitioning. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(2):1–24.
- 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.
- Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. *arXiv preprint arXiv:2104.07302*.
- Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.
- Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.
- Jiahuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.
- Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.
- Bayu Distiawan Trisedya, Flora D Salim, Jeffrey Chan, Damiano Spina, Falk Scholer, and Mark Sanderson. 2023. i-align: an interpretable knowledge graph alignment model. *Data Mining and Knowledge Discovery*, 37(6):2494–2516.
- Kaehyun Um, KyuHwan Yeom, Haerim Yang, Minyoung Choi, Hyeongjun Yang, and Kyong-Ho Lee. 2026. Rpo-rag: Aligning small llms with relation-aware preference optimization for knowledge graph question answering. In *Proceedings of the ACM Web Conference 2026*, pages 2513–2523.
- Haoyuan Wu, Zhuolun He, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, and Bei Yu. 2024. Chateda: A large language model powered autonomous agent for eda. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.
- Yuchen Xia, Manthan Shenoy, Nasser Jazdi, and Michael Weyrich. 2023. Towards autonomous system: flexible modular production system enhanced with large language model agents. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE.
- Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.
- Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey. *arXiv preprint arXiv:2405.07437*.
- Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM Web Conference 2024*, pages 3679–3689.
- Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Chen Xu, Yulong Chen, Longyue Wang,

Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2025. [Siren’s song in the ai ocean: A survey on hallucination in large language models](#). *Preprint*, arXiv:2309.01219.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. 2023. Competeai: Understanding the competition dynamics in large language model-based agents. *arXiv preprint arXiv:2310.17512*.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

A Notations

Important notations are listed in [Table 9](#).

B Baseline Description

Detailed descriptions of the baselines are provided in [Table 11](#).

C Example of how training questions are preprocessed and stored

Examples of different formats of questions are presented in [Table 10](#). Beside raw question, semantic context, entity-type context and path context are used in our experiments.

D Algorithms

Algorithms [1](#) to [5](#) provide the implementation procedures for the main components of SPLIT-RAG.

Table 9: Important notations used in SPLIT-RAG.

Notation	Definition
F	The SPLIT-RAG framework.
$\hat{G}$	Multi-agent generator for final answer synthesis.
D	Original knowledge graph.
τ	Offline indexer that constructs graph partitions from D and Q_{train} .
ψ	Query-time retriever that selects evidence from activated partitions.
$\hat{K}$	Partitioned knowledge base constructed from D .
Q_{train}	Training-question set used to guide graph partitioning and routing.
$\mathbf{Q}_e$	Entity-type representation of a question.
$\mathbf{p}$	Reasoning path associated with a question.
$\mathcal{S}$	Candidate graph partition.
$\mathcal{C}$	Question-pattern cluster variable.
Z	Partition assignment variable of a reasoning-path fragment.
$\hat{D}_i$	The i -th graph partition.
$\mathcal{G}_\ell$	Set of partitions assigned to agent a_ℓ .
$\mathcal{A}^*(q)$	Set of agents activated for question q .
$\mathcal{T}_i$	Structured triples retrieved by agent a_i .
$\mathcal{E}_i$	Textualised evidence returned by agent a_i .
$\mathcal{T}_{\text{clean}}$	Compatible evidence set retained for final answer generation.

Table 10: Illustrative example of the processed representations of a training question stored in the question base.

Question Format	Example
Raw Question	Which genres contain films that share directors with [Black Snake Moan]?
Semantic Context	films share directors film [Black Snake Moan] genres
Entity-Type Context	movie share director movie genre
Path Context	movie-[directed_by]-director-[directed_by]-{Black Snake Moan}-{has_genre}-genre

Table 11: Descriptions of the baselines used in the experiments.

Type	Method	Description
Embedding	KV-Mem (Miller et al., 2016)	Stores knowledge triples in a key-value memory network for multi-hop reasoning.
	GraftNet (Sun et al., 2018)	Performs reasoning over question-specific KG subgraphs.
	PullNet (Sun et al., 2019)	Iteratively retrieves question-relevant subgraphs using graph neural networks.
	EmbedKGQA (Saxena et al., 2020)	Formulates KGQA as link prediction using learned entity and relation embeddings.
	TransferNet (Shi et al., 2021)	Propagates relevance scores over relation graphs for multi-hop answering.
KG+LLM	Graph-RAG	Standard KG-based retrieval-augmented generation baseline.
	RoG (Luo et al., 2023)	Uses planning, graph retrieval, and reasoning for KG-based question answering.
	SubgraphRAG (Li et al., 2024a)	Retrieves compact question-relevant KG subgraphs and uses them as context for LLM reasoning.
	ToG (Sun et al., 2023)	Performs iterative LLM-guided graph exploration to construct reasoning paths over the KG.
	GcR (Luo et al., 2024)	Constrains LLM reasoning with graph-derived evidence to improve faithfulness in KGQA.
	GNN-RAG (Mavromatis and Karypis, 2025)	Uses graph-neural retrieval to select compact reasoning evidence for LLM-based KGQA.
	RPO-RAG (Um et al., 2026)	Uses relation-aware preference optimisation to improve small-LLM reasoning over retrieved KG paths.
	SPLIT-RAG	Uses question-guided graph partitioning with routed specialised retrieval agents.

Algorithm 1 Question-Guided KG Partitioning

Require: KG $D = (\mathcal{V}, \mathcal{E})$; training path fragments $\tilde{\mathcal{P}}$; question-pattern labels $\mathcal{C}$; $\lambda, \mu, \eta_{\max}$
Ensure: Partitioned knowledge base $\hat{\mathcal{K}} = \{\hat{D}_1, \dots, \hat{D}_M\}$

- 1: Initialise seed regions $\mathcal{S}$ from fragments in $\tilde{\mathcal{P}}$
- 2: **repeat**
- 3: $(s_a, s_b) \leftarrow \arg \max_{s_a, s_b \in \mathcal{S}} \Delta(s_a, s_b)$,
 where
$$\Delta(s_a, s_b) = \mathcal{J}(\mathcal{S} \setminus \{s_a, s_b\} \cup \{s_a \cup s_b\}) - \mathcal{J}(\mathcal{S})$$
- 4: **if** $\Delta(s_a, s_b) > 0$ **and** $|\mathcal{V}_{s_a \cup s_b}| \leq \eta_{\max}$ **then**
- 5: $\mathcal{S} \leftarrow \mathcal{S} \setminus \{s_a, s_b\} \cup \{s_a \cup s_b\}$
- 6: **else**
- 7: **break**
- 8: **end if**
- 9: **until** no admissible merge improves Eq. (3)
- 10: **for** each uncovered triple $\tau \in \mathcal{E}$ **do**
- 11: Assign τ to its nearest compatible region in $\mathcal{S}$
- 12: **end for**
- 13: Retain boundary entities required by cross-partition paths
- 14: **return** $\hat{\mathcal{K}} \leftarrow \{\hat{D}_1, \dots, \hat{D}_M\}$

Algorithm 2: Question-Centric Agent-Subgraph Allocation

Require: Partitions $\hat{\mathcal{K}} = \{\hat{D}_1, \dots, \hat{D}_M\}$; training questions $\mathcal{Q}_{\text{train}}$; capacity $N_{\max}$; coherence threshold θ_{coh}
Ensure: Agent groups $\{\mathcal{G}_1, \dots, \mathcal{G}_L\}$

- 1: Construct coverage sets $\mathcal{C}_i$ via Eq. (6)
- 2: Initialise priority queue $\mathcal{PQ}$ with
$$\rho(\mathcal{C}_i) = \sum_j |\mathcal{C}_i \cap \mathcal{C}_j| / \sqrt{|\mathcal{C}_i|}$$
- 3: Initialise unassigned partitions $\mathcal{U} \leftarrow \hat{\mathcal{K}}, L \leftarrow 0$
- 4: **while** $\mathcal{PQ} \neq \emptyset$ **and** $\mathcal{U} \neq \emptyset$ **do**
- 5: $\mathcal{C}^* \leftarrow \arg \max_{\mathcal{C} \in \mathcal{PQ}} \rho(\mathcal{C})$
- 6: $\mathcal{G}_{\text{cand}} \leftarrow \mathcal{C}^* \cap \mathcal{U}$
- 7: **if** $|\mathcal{G}_{\text{cand}}| > N_{\max}$ **then**
- 8: Retain the top- $N_{\max}$ partitions ranked by
$$f(\hat{D}_j) = \sum_i \mathbb{I}[\hat{D}_j \in \mathcal{C}_i]$$
- 9: **end if**
- 10: **if** $\text{Coh}(\mathcal{G}_{\text{cand}}) \geq \theta_{\text{coh}}$ **then**
- 11: $L \leftarrow L + 1; \mathcal{G}_L \leftarrow \mathcal{G}_{\text{cand}}$
- 12: $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{G}_L$
- 13: Remove covered candidate sets from $\mathcal{PQ}$
- 14: **else**

- 15: Remove $\mathcal{C}^*$ from $\mathcal{PQ}$
- 16: **end if**
- 17: **end while**
- 18: **for** each $\hat{D}_j \in \mathcal{U}$ **do**
- 19: $\ell^* \leftarrow \arg \max_{\ell: |\mathcal{G}_\ell| < N_{\max}} \text{Sim}(\hat{D}_j, \mu_\ell)$
- 20: $\mathcal{G}_{\ell^*} \leftarrow \mathcal{G}_{\ell^*} \cup \{\hat{D}_j\}$
- 21: **end for**
- 22: **return** $\{\mathcal{G}_1, \dots, \mathcal{G}_L\}$

Algorithm 3 Query-Time Agent Routing

Require: Question q ; training registry $\mathcal{M}$; partition-agent groups $\{\mathcal{G}_\ell\}_{\ell=1}^L$
Ensure: Activated agent-task mapping $\mathcal{MAP} = \{(a_\ell, t_\ell)\}$

- 1: Form the entity-type representation $\mathbf{Q}_{e,q}$
- 2: Predict relation-path sketch $\hat{\mathbf{p}}_q$ from q without using gold test paths or answers
- 3: Compute $\text{Sim}(q, q_i)$ using Eq. (9)
- 4: Retrieve neighbours $\mathcal{N}_\theta(q)$
- 5: Construct $\mathcal{A}_{\text{sim}}(q)$ using Eq. (10)
- 6: Construct $\mathcal{A}_{\text{path}}(q)$ using Eq. (11)
- 7: Determine $d(q)$ using threshold θ_{direct}
- 8: Activate $\mathcal{A}^*(q)$ using Eq. (12)
- 9: **if** $d(q) = 1$ **then**
- 10: Transfer sub-task patterns from the nearest matched training questions
- 11: **else**
- 12: Construct sub-tasks from fragments in $\phi(\hat{\mathbf{p}}_q)$
- 13: **end if**
- 14: **for** each activated agent $a_\ell \in \mathcal{A}^*(q)$ **do**
- 15: Assign its aligned sub-task t_ℓ to $\mathcal{MAP}$
- 16: **end for**
- 17: **return** $\mathcal{MAP}$

Algorithm 4 Parallel Subgraph Retrieval for Agent a_i

Require: Sub-task t_i ; agent a_i ; assigned partitions $\mathcal{G}_{a_i}$
Ensure: Structured triples $\mathcal{T}_i$; textual evidence $\mathcal{E}_i$

- 1: Retrieve matched paths:
$$\mathcal{P}_i \leftarrow \{p \in \text{PATHS}(\mathcal{G}_{a_i}) : \text{Match}(p, t_i) \geq \theta_{\text{match}}\}$$

 ▷ using Eq. (13)
- 2: Extract triples:
$$\mathcal{T}_i \leftarrow \bigcup_{p \in \mathcal{P}_i} \{\tau : \tau \in p\}$$
- 3: Textualise retrieved triples:
$$\mathcal{E}_i \leftarrow \{\text{TEXTUALIZE}(\tau) : \tau \in \mathcal{T}_i\}$$
- 4: **return** $(\mathcal{T}_i, \mathcal{E}_i)$

Algorithm 5 Compatibility-Based Evidence Filtering

Require: Retrieved triples $\mathcal{T}_{\text{all}}$; textual evidence $\mathcal{E}_{\text{all}}$; activated agents $\mathcal{A}^*(q)$; functional relations $\mathcal{R}_{\text{func}}$

Ensure: Filtered triple set $\mathcal{T}_{\text{clean}}$

```

1: for each  $\tau \in \mathcal{T}_{\text{all}}$  do
2:   Compute  $w(\tau)$  using Eq. (14)
3: end for
4: Sort  $\mathcal{T}_{\text{all}}$  in descending order of  $w(\tau)$ 
5: Break ties by  $\text{Freq}(\tau)$ ,  $\text{Conf}(a_i)$ , then lexical order
6:  $\mathcal{T}_{\text{clean}} \leftarrow \emptyset$ 
7: for each  $\tau \in \mathcal{T}_{\text{all}}$  in sorted order do
8:   if  $\text{Conflict}(\tau, \tau') = 0, \forall \tau' \in \mathcal{T}_{\text{clean}}$  then
9:      $\mathcal{T}_{\text{clean}} \leftarrow \mathcal{T}_{\text{clean}} \cup \{\tau\}$ 
10:  end if
11: end for
12: return  $\mathcal{T}_{\text{clean}}$ 

```

E Analysis of Partition Alignment and Retrieval Efficiency

This section provides conditional properties of SPLIT-RAG. The analysis does not assume that routing perfectly covers every unseen question. Instead, it characterises (i) what is encouraged by the partition objective, (ii) when routing preserves sufficient evidence, and (iii) when selective activation reduces retrieval cost.

Proposition 1 (Question-Pattern Alignment Objective). *Let C denote the question-pattern cluster associated with a sampled training reasoning fragment, and let Z denote the partition assignment of that fragment under a candidate partition $\mathcal{S}$. For the partition objective in Eq. (3),*

$$\mathcal{J}(\mathcal{S}) = I(C; Z) - \lambda \Omega_{\text{size}}(\mathcal{S}) - \mu |\mathcal{S}|, \quad (18)$$

with $\lambda, \mu \geq 0$, the induced mutual information satisfies

$$I(C; Z) = \mathcal{J}(\mathcal{S}) + \lambda \Omega_{\text{size}}(\mathcal{S}) + \mu |\mathcal{S}| \geq \mathcal{J}(\mathcal{S}). \quad (19)$$

Therefore, subject to the size and fragmentation penalties, maximising $\mathcal{J}(\mathcal{S})$ favours partitions whose assignments are informative of question-pattern clusters.

Proof. By definition of the objective in Eq. (3),

$$\mathcal{J}(\mathcal{S}) = I(C; Z) - \lambda \Omega_{\text{size}}(\mathcal{S}) - \mu |\mathcal{S}|. \quad (20)$$

Rearranging gives

$$I(C; Z) = \mathcal{J}(\mathcal{S}) + \lambda \Omega_{\text{size}}(\mathcal{S}) + \mu |\mathcal{S}|. \quad (21)$$

Since $\lambda, \mu \geq 0$, $\Omega_{\text{size}}(\mathcal{S}) \geq 0$, and $|\mathcal{S}| \geq 0$, it follows that

$$I(C; Z) \geq \mathcal{J}(\mathcal{S}). \quad (22)$$

Thus, after accounting for compactness and fragmentation penalties, improving the objective encourages stronger dependence between question-pattern clusters and partition assignments. $\square$

Proposition 2 (Evidence Preservation under Routing Coverage). *For a question q , let $\mathcal{T}^*(q) \subseteq \mathcal{E}$ be a sufficient set of KG triples for deriving a correct answer. Let*

$$\mathcal{U}(q) = \bigcup_{a_i \in \mathcal{A}^*(q)} \mathcal{G}_{a_i} \quad (23)$$

denote the union of partitions activated by the router. If

$$\mathcal{T}^*(q) \subseteq \mathcal{U}(q), \quad (24)$$

and the local retrievers recover all triples in $\mathcal{T}^*(q)$ contained in $\mathcal{U}(q)$, then SPLIT-RAG preserves the supporting evidence required for answering q relative to retrieval over the full graph.

Proof. By the routing coverage condition in Eq. (24), every triple required by the sufficient evidence set $\mathcal{T}^*(q)$ lies inside the union of activated partitions $\mathcal{U}(q)$. By assumption, the local retrievers recover every required triple from $\mathcal{U}(q)$. Therefore,

$$\mathcal{T}^*(q) \subseteq \mathcal{T}_{\text{all}}, \quad (25)$$

where $\mathcal{T}_{\text{all}}$ is the retrieved evidence supplied to the aggregation module. Hence, routing does not remove any triple necessary for deriving the answer from $\mathcal{T}^*(q)$. $\square$

Proposition 3 (Retrieval-Cost Reduction under Selective Activation). *Let $n = |\mathcal{E}|$ be the number of triples in the full KG, and let $n_q = |\mathcal{E}(\mathcal{U}(q))|$ be the number of triples contained in the partitions activated for question q . Let $C(x)$ be a monotone retrieval-cost function for searching over x triples. If $T_{\text{plan}}(q)$ and $T_{\text{agg}}(q)$ denote routing and aggregation overhead, respectively, then*

$$\begin{aligned} T_{\text{split}}(q) &= T_{\text{plan}}(q) + C(n_q) + T_{\text{agg}}(q), \\ T_{\text{full}}(q) &= C(n). \end{aligned} \quad (26)$$

Consequently, SPLIT-RAG reduces online retrieval time whenever

$$T_{\text{plan}}(q) + C(n_q) + T_{\text{agg}}(q) < C(n). \quad (27)$$

Proof. Full-graph retrieval searches over all n triples and therefore incurs cost $C(n)$. SPLIT-RAG first constructs a query-specific routing plan, retrieves only from the activated graph union containing n_q triples, and aggregates the resulting evidence. Thus, its total online cost is

$$T_{\text{split}}(q) = T_{\text{plan}}(q) + C(n_q) + T_{\text{agg}}(q). \quad (28)$$

Comparing this expression with $T_{\text{full}}(q) = C(n)$ immediately yields Eq. (27). Therefore, the efficiency gain is determined by the reduction in activated graph size relative to routing and aggregation overhead. $\square$

Parallel-agent execution. When activated agents retrieve evidence in parallel, let $n_{q,\ell}$ be the number of triples searched by agent a_ℓ . Ignoring communication overhead already included in $T_{\text{agg}}(q)$, the retrieval component of wall-clock latency becomes

$$C_{\text{parallel}}(q) = \max_{a_\ell \in \mathcal{A}^*(q)} C(n_{q,\ell}), \quad (29)$$

rather than the sum of per-agent retrieval costs. Accordingly, the parallel execution condition is

$$T_{\text{plan}}(q) + \max_{a_\ell \in \mathcal{A}^*(q)} C(n_{q,\ell}) + T_{\text{agg}}(q) < C(n). \quad (30)$$

This condition motivates the latency evaluation reported in the main experiments.

Additional notes on fairness. All runs use the fixed decoding and retrieval budgets for MetaQA2-hop and 3-hop are in Table 14. While for WebQSP and CWQ datasets the results are recorded in Table 13. The wall-clock time is measured end-to-end per QA, averaging over the MetaQA-3Hop test split.

F Path-Sketch Prediction and Routing Details

Relation-path sketch prediction. For each test question q , SPLIT-RAG predicts a relation-path sketch $\hat{\mathbf{p}}_q$ before agent activation. The predictor receives only the natural-language question, linked seed entities, and the relation vocabulary available in the indexed KG; it does not access gold answers or gold reasoning paths. We implement the predictor using Llama-3.1-8B. The predictor is prompted to produce an ordered sequence of at most H_{max} relation labels, where H_{max} is set to the maximum reasoning depth of the corresponding

benchmark. Decoding uses temperature 0.1, top- p 0.9, and retains the top- K 3 path sketches. The cost of sketch prediction is included in all reported end-to-end latency measurements.

Relation vocabulary. The candidate vocabulary is constructed from relation labels appearing in the indexed knowledge graph after preprocessing. For efficiency, candidate relations may be restricted to those adjacent to linked seed entities and those observed in training reasoning paths associated with the retrieved question-pattern neighbours. No relation labels derived from gold test paths are used.

Path-overlap score. Let $\text{Rel}(p)$ denote the ordered relation sequence of path p , and let $\text{LCS}(\cdot, \cdot)$ denote the length of their longest common subsequence. We define the path-overlap component in Eq. (9) as

$$\text{PathOverlap}(\hat{\mathbf{p}}_q, \mathbf{p}_i) = \frac{\text{LCS}(\text{Rel}(\hat{\mathbf{p}}_q), \text{Rel}(\mathbf{p}_i))}{\max\{1, |\text{Rel}(\hat{\mathbf{p}}_q)|\}}. \quad (31)$$

This score rewards relation-order agreement while allowing a predicted sketch to partially match a longer training path.

Partition compatibility. For a predicted path fragment $\tilde{p}$ and a partition $\hat{D}_j$, we define

$$\text{Compatible}(\tilde{p}, \hat{D}_j) = \mathbb{I} \left[\max_{p \in \mathcal{P}(\hat{D}_j)} \text{PathOverlap}(\tilde{p}, p) \geq \theta_{\text{path}} \right], \quad (32)$$

where $\mathcal{P}(\hat{D}_j)$ is the set of indexed relation-path fragments associated with partition $\hat{D}_j$. We set $\theta_{\text{path}} = 0.5$ on the validation split and keep it fixed for test evaluation.

Routing without path sketches. When no relation-path sketch is available, or when its prediction confidence is below θ_{pred} , the path term is disabled by setting $\gamma_p = 0$, and routing relies only on semantic similarity between the incoming question and stored training-question patterns. The contribution of predicted path sketches is evaluated in Table 12.

G Ablation

G.1 Additional Ablation Diagnostics

To further interpret the component ablations in Table 6, we compute two diagnostic statistics on WebQSP. For variant B, a routed retrieval region is counted as disconnected if the union of its selected partitions contains more than one connected component after merging. Under this defini-

Table 12: Ablation of routing signals on WebQSP. Semantic denotes entity-type question similarity; Path denotes the predicted relation-path sketch.

Routing Signal	Hit	H@1	F1	Coverage	Time (s)
Semantic only ($\gamma_p = 0$)	90.6	86.9	79.7	88.4	29.8
Path only ($\gamma_s = 0$)	88.1	83.8	76.5	82.0	30.5
Semantic + predicted path	92.8	89.2	82.2	91.3	31.6

tion, 22.3% of evaluated questions produce disconnected merged regions. For variant C, a question is counted as containing an explicit incompatibility if at least one pair of retrieved triples satisfies $\text{Conflict}(\tau_1, \tau_2) = 1$ before compatibility-based filtering. Explicit incompatibilities are detected in 12.3% of evaluated questions.

H Evidence Aggregation Details

Support-score components. For each retrieved triple τ , SPLIT-RAG computes the support score in Eq. (14) using three normalised components. The agent-confidence term is estimated from training-time retrieval success:

$$\text{Conf}(a_i) = \frac{n_i^{\text{succ}} + 1}{n_i^{\text{ret}} + 2}, \quad (33)$$

where n_i^{ret} is the number of training questions routed to agent a_i , and n_i^{succ} is the number for which the agent retrieves at least one gold supporting triple. The additive smoothing prevents unreliable confidence estimates for rarely activated agents.

The multi-agent frequency component is defined as

$$\text{Freq}(\tau) = \frac{\sum_{a_i \in \mathcal{A}^*(q)} \mathbb{I}[\tau \in \mathcal{T}_i]}{|\mathcal{A}^*(q)|}. \quad (34)$$

It rewards triples independently recovered from multiple activated agents.

Evidence entailment verifier. The entailment term evaluates whether a retrieved triple is supported by the textualised evidence collected from activated agents:

$$\text{Entail}(\mathcal{E}_{\text{all}}, \tau) = P_{\text{ver}}(\text{SUPPORTED} \mid \mathcal{E}_{\text{all}}, \tau). \quad (35)$$

We implement P_{ver} using Llama-3.1-8b with temperature 0.0. The verifier receives the candidate triple and its associated textualised evidence and returns a scalar support score in $[0, 1]$.

Aggregation weights. The coefficients in Eq. (14) are selected on the validation split and

fixed for all test questions:

$$\alpha = 0.5, \quad \beta = 0.25, \quad \gamma = 0.25. \quad (36)$$

We tune these coefficients over $0, 0.25, 0.5, 0.75, 1.0$ using validation-set H@1 as the selection criterion.

Functional relations. To avoid treating valid multi-valued relations as contradictions, object collisions are considered conflicts only for relations estimated to be functional. For each relation r , let $\text{deg}_r^+(e)$ denote the number of distinct objects linked from entity e through r . We construct the functional-relation set as

$$\mathcal{R}_{\text{func}} = \left\{ r : \frac{|\{e: \text{deg}_r^+(e) \leq 1 \wedge \text{deg}_r^+(e) > 0\}|}{|\{e: \text{deg}_r^+(e) > 0\}|} \geq \theta_{\text{func}} \right\}. \quad (37)$$

The threshold $\theta_{\text{func}} = 0.85$ is set using the training KG only and is not derived from test answers. For two triples $\tau_1 = (e, r, o_1)$ and $\tau_2 = (e, r, o_2)$, functional collision is defined as

$$\text{FuncCol}(\tau_1, \tau_2) = \mathbb{I}[r \in \mathcal{R}_{\text{func}} \wedge o_1 \neq o_2]. \quad (38)$$

Greedy compatible-set construction. The retrieved triples are first sorted by support score $w(\tau)$ in descending order. Ties are broken deterministically by $\text{Freq}(\tau)$, then by $\text{Conf}(a_i)$, and finally by lexical triple order. A triple is retained only if it is compatible with every previously retained triple. Algorithm 5 provides the complete procedure.

I Experiment Data and Prompt Template

Table 13: Fixed decoding & retrieval budgets (WebQSP / CWQ).

Setting	$\{\mathcal{A}_i\}$ (subgraph agents)	$\mathbf{A}_{\text{head}}$
Max input (tokens) per call	3,000	4,096
Max new tokens	96	256
Temperature / Top- p	0.2 / 0.9	0.3 / 0.95
Stop sequences	$\backslash n \backslash n$, KB-EOF	$\backslash n \backslash n$, Answer-EOF
Tool-calls / function-calls	disabled	disabled
Evidence cap to head (tokens)	—	3,200
Max subgraphs per question B	4	4
Path depth / top- K per subgraph	depth ≤ 2 , top- $K = 8$	—
Max triplets per agent	600	—

I.1 Experiment compute resources

Graph-partitioning and local inference with Llama-3-8B were executed on four NVIDIA RTX A5000 GPUs (24 GB VRAM each). Calls to Gemini and GPT were made through remote APIs, so no local GPU was required; a CPU-only setup was sufficient. All experiments ran on a server equipped with an Intel Xeon Silver 4310 (12 cores, 2.10 GHz) processor.

Table 14: Fixed decoding & retrieval budgets (MetaQA 3-Hop).

Setting	$\{A_i\}$ (subgraph agents)	A_{head}
Max input (tokens) per call	3,000	4,096
Max new tokens	96	256
Temperature / Top- p	0.2 / 0.9	0.3 / 0.95
Stop sequences	$\backslash n \backslash n$, KB-EOF	$\backslash n \backslash n$, Answer-EOF
Tool-calls / function-calls	disabled	disabled
Evidence cap to head (tokens)	—	2,600
Max subgraphs per question B	3	3
Path depth / top- K per subgraph	$\text{depth} \leq 2$, top- $K = 5$	—
Max triplets per agent	400	—

I.2 Prompt Template

I.2.1 Subgraph Control Agent $\mathcal{A}_i$ Prompt

```

Given Subgraph Context:
- Entity Types: {<T_entity>}
- Relation Types: {<T_relations>}
- Coverage: This subgraph focuses on
  ↳ <label> relationships
Given Current Subquestion: <Q_sub>

Task:
1. Analyze the subquestion's core
  ↳ information need
2. Generate {SPARQL/Cypher} query
  ↳ matching the subgraph schema

Critical Constraints:
- Use ONLY entities/relations from the
  ↳ subgraph context
- Query returns triples that directly
  ↳ answer the subquestion

Output Requirements:
{
  "query": "<generated_query>",
  "reasoning": "<brief_explanation_of_
  ↳ _strategy>"
}

```

I.2.2 Head Agent $\mathcal{A}_H$ Prompt

```

Given verified facts: <T_clean>
Supporting evidence: <E_all>
Original question: <q_new>

Generate final answer with
  ↳ explanations,
resolving any remaining ambiguities.

```

I.2.3 Relation-Path Sketch Predictor Prompt

```

Given a knowledge-graph question and
  ↳ its linked seed entities,
predict a short relation-path sketch
  ↳ that is likely to connect
the seed entities to the answer.

Question:
<Q>

Linked Seed Entities:
<E_seed>

Candidate Relation Vocabulary:
<R_candidate>

```

Constraints:

1. Use only relations from the
↳ candidate relation vocabulary.
2. Produce at most <H_max> relations.
3. Do not generate answer entities.
4. Return relation paths only, ranked
↳ from most to least likely.

Output Format:

```

{
  "paths": [
    {
      "relations": ["<r_1>", "<r_2>",
        ↳ "..."],
      "confidence": <score>
    }
  ]
}

```

J Use of AI Assistants

We used a conversational AI assistant for help in writing and editing manuscript; no scientific and numerical claim in this paper was generated by an AI assistant.