When Should We Build the Graph? An Amortized Pareto View of Demand-Driven Knowledge Graph Construction

Niharika S. D'Souza[1], Satyananda Kashyap[1], Nandana Mihindukulasooriya[2], Jui-Chien Lin[3], Oshani Seneviratne[3] and Horst Samulowitz[2]

[1]IBM Silicon Valley Labs
[2]IBM Yorktown Heights
[3]Rennselaer Polytechnic Institute

Abstract

Knowledge graph construction for GraphRAG has traditionally been organized around documents: graphs are materialized offline over a corpus, or updated as new documents arrive. Many retrieval workloads instead reveal information needs only through a query stream, raising a complementary question: *when should graph structure be constructed?* We introduce *demand-driven graph construction*, in which each query triggers materialization of only the relevant semantic neighborhood; that structure persists in a shared graph and is reused by later queries, so growth is driven by demand rather than document arrival. This formulation treats graph construction as an amortized optimization problem balancing semantic construction cost, graph reuse, and downstream QA quality rather than maximizing graph completeness upfront. We formalize the construction policy and its amortized objective, instantiate it using a pay-as-you-go entity knowledge graph, and introduce workload-level measures that characterize graph growth and stabilization over persistent query streams. Experiments on MuSiQue, HotpotQA, and 2WikiMultiHopQA demonstrate that demand-driven graph construction consistently approaches the performance of fully materialized knowledge graphs while substantially reducing semantic construction cost. We further show that explicit relation extraction is essential for realizing this quality–cost frontier and that the preferred operating point depends on the reasoning characteristics of the workload. These results suggest that GraphRAG systems should be evaluated not only by downstream accuracy, but also by when and where semantic structure is constructed under evolving demand.

Keywords

demand-driven knowledge graph construction, GraphRAG, retrieval-augmented generation, amortized graph construction, multi-hop question answering, knowledge graph construction

CEUR-WS.org/Vol-4335/paper5.pdf

1. Introduction

Knowledge graphs (KGs) have become a key retrieval substrate for GraphRAG, providing structured representations that support multi-hop reasoning, interpretable retrieval, and evidence aggregation over unstructured corpora [1]. Existing work has largely focused on improving *how* graphs are constructed through advances in entity extraction, relation extraction, schema induction, graph integration, and quality assurance [2]. Whether constructed offline over an entire corpus or incrementally as new documents arrive, these approaches share a common assumption: semantic structure is materialized before downstream retrieval begins.

This assumption is increasingly mismatched to modern retrieval workloads. Enterprise assistants, scientific copilots, and agentic systems operate over large, continuously evolving corpora where only a small fraction of documents may ever contribute to answering user queries. Fully materializing a knowledge graph therefore performs expensive semantic extraction over regions of the corpus that may never be used. Document-driven incremental construction only partly avoids this cost: new documents are still extracted on arrival even if no later query needs them. This raises a simple but largely unexplored question: *When should we build the graph?* We argue that KG construction can be categorized according to *what triggers graph materialization* (Table 1). Traditional approaches are *data-driven*, constructing graphs once a corpus becomes available. Incremental approaches are *document-driven*, updating an

5th International Workshop on Natural Language Processing for Knowledge Graph Construction (NLP4KGC 2026), co-located with ISWC 2026, October 26, 2026, Bari, Italy



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-19

Aspect	Data-driven	Document-driven	Demand-driven (Ours)
Trigger	Corpus	New documents	User queries
Construction scope	Entire corpus	New documents merged into an existing graph	Query-relevant neighborhood appended to a persistent graph
Construction timing	Before querying	During ingestion	During retrieval
Optimization objective	Graph completeness	Graph maintenance	Quality–cost–reuse

Table 1
Knowledge graph construction paradigms categorized by what triggers graph construction.

existing graph as new documents arrive. We instead introduce *demand-driven graph construction*, where user queries provide the demand signal and retrieval identifies source evidence for incremental graph materialization.

Demand-driven graph construction treats semantic extraction as an incremental investment rather than a fixed preprocessing step. Instead of materializing an entire corpus offline, graph structure is constructed only for passages required to answer observed queries and is subsequently reused by future queries. The graph therefore evolves as a persistent semantic cache whose growth is driven by information demand. This changes the optimization objective from maximizing graph completeness to balancing downstream QA quality against semantic construction and serving costs, naturally giving rise to an *amortized Pareto view* of KG construction. To study this paradigm, we instantiate a simple pay-as-you-go construction policy. For each query, the top-K retrieved source chunks are considered for admission. A chunk is admitted if it has not previously been materialized; admitted chunks are processed by the entity–relation extractor, while previously materialized chunks are reused. Thus, the current policy detects *source-chunk novelty* rather than explicitly determining whether the knowledge relevant to the query is absent from the graph. Previously unseen passages are processed by an entity–relation extractor and appended to a persistent entity knowledge graph, while previously materialized regions are reused. A graph retriever then performs Personalized PageRank over the evolving graph before a fixed reader generates the final answer. Throughout the study, the retriever, extractor, graph retriever, and reader remain fixed, allowing us to isolate the effect of construction timing rather than architectural choices.

We evaluate the proposed framework on MuSiQue [3], whose compositional 2-, 3-, and 4-hop questions provide a challenging setting for demand-driven graph construction. We further evaluate HotpotQA [4] and 2WikiMultiHopQA [5] to assess generalization across diverse multi-hop QA benchmarks. Compared with fully materialized knowledge graphs, amortized text-based retrieval, and relation-free graph baselines, our experiments characterize the quality–cost frontier of pay-as-you-go KG construction and identify when semantic graph construction can be deferred without sacrificing downstream QA quality. Our contributions are summarized as follows.

1. We introduce *demand-driven graph construction* as a distinct operational paradigm for GraphRAG, complementing traditional data-driven and document-driven knowledge graph construction.

2. We formulate an amortized Pareto view of KG construction that jointly considers graph construction cost, serving cost, graph growth, and downstream QA quality.

3. We introduce construction-aware evaluation metrics that characterize graph growth, graph reuse, and stabilization under persistent query workloads.

4. We perform the first systematic empirical study of demand-driven graph construction across MuSiQue, HotpotQA, and 2WikiMultiHopQA, showing that substantial portions of offline graph construction can be deferred while preserving downstream QA performance.

5. We demonstrate that explicit relation extraction is essential for realizing the quality–cost frontier through comparisons against relation-free graph construction baselines.

Traditional approaches are *data-driven*, constructing graphs once a corpus becomes available. Incremental approaches are *document-driven*, updating an existing graph as new documents arrive. We instead introduce *demand-driven graph construction*, where user queries provide the demand signal and retrieval identifies source evidence for incremental graph materialization.

2. Related Work

Knowledge graph construction. Knowledge graph construction has traditionally been formulated as a semantic extraction problem performed before downstream retrieval or reasoning. Recent LLM-based approaches have substantially improved entity extraction, relation extraction, schema induction, ontology generation, and graph refinement, with systems such as AutoSchemaKG jointly inducing schemas and constructing knowledge graphs for downstream reasoning [6]. Recent surveys identify extraction quality, graph integration, schema evolution, and quality assurance as central research challenges [2]. Existing work primarily optimizes *how* semantic structure is constructed; we instead study *when* that structure should be materialized for a query workload.

Incremental knowledge graph construction. As new documents arrive, incremental KG construction updates an existing graph through incremental extraction, entity resolution, and graph integration. Systems such as iText2KG update knowledge graphs continuously while preserving graph consistency [7]. These approaches are driven by *data arrival*: newly available documents trigger graph updates. In contrast, we assume a fixed corpus and study graph expansion driven by user queries, treating semantic materialization as a scheduling problem over information demand.

GraphRAG and graph retrieval. Knowledge graphs have become an effective retrieval substrate for multi-hop reasoning in GraphRAG [1]. Recent work has explored query-driven graph induction [8] and persistent graph memories such as HippoRAG [9] to improve graph-based retrieval over evolving workloads. These approaches primarily focus on constructing or retrieving graph representations to support downstream reasoning. Our setting is complementary: queries provide the demand signal, retrieval identifies candidate source evidence, and previously unseen evidence is incrementally materialized into a persistent semantic graph.

Pay-as-you-go semantic integration. Our formulation is closely related to pay-as-you-go semantic integration, where semantic structure evolves incrementally rather than through complete upfront integration [10]. Sequeda et al. [11] introduced a pay-as-you-go methodology for constructing enterprise knowledge graphs from large relational databases, progressively engineering ontologies and mappings in response to business information needs rather than requiring complete semantic integration upfront. Lifelong extraction systems such as NELL similarly demonstrated that structured knowledge can accumulate through repeated interaction with large corpora [12]. We extend these ideas to GraphRAG by introducing *demand-driven graph construction*, where user queries provide the demand signal for incremental semantic materialization and graph growth is evaluated jointly under construction cost, reuse, and downstream QA quality.

Taken together, existing work has primarily addressed graph construction quality, graph maintenance, semantic integration, or graph-based retrieval. In contrast, we study the *timing of semantic materialization* itself: rather than constructing the full graph before retrieval or updating it when data arrives, semantic structure is progressively materialized from query-relevant source evidence and reused across a persistent query workload.

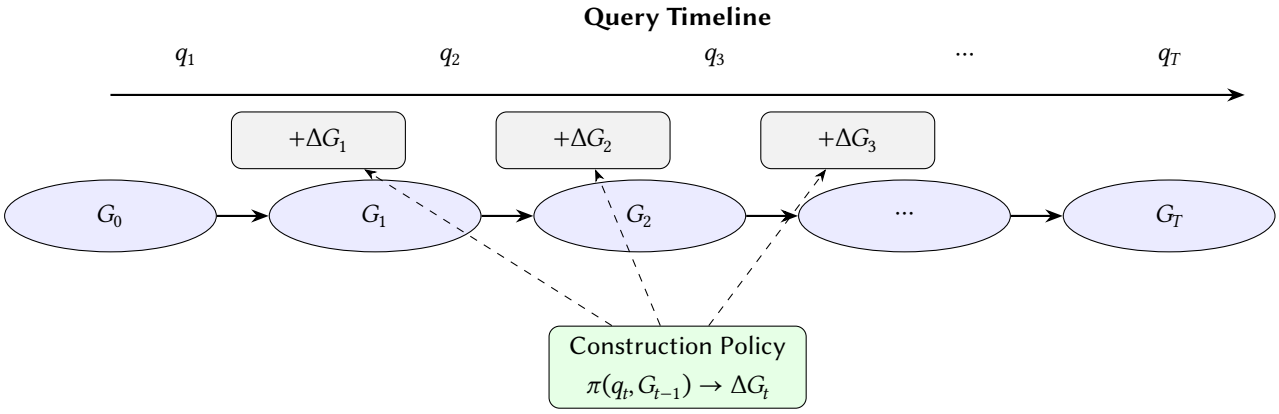


Figure 1: Demand-driven knowledge graph construction. Rather than materializing a complete knowledge graph before deployment, a persistent graph evolves jointly with the query stream. For each incoming query, a construction policy determines the previously unseen semantic information (ΔG_t) to materialize, yielding the graph update $G_t = G_{t-1} \cup \Delta G_t$. Previously constructed graph regions persist across queries, allowing graph construction cost to be amortized over time.

3. Demand-Driven Knowledge Graph Construction

Traditional knowledge graph construction assumes that semantic structure is materialized before downstream retrieval begins or incrementally as new documents arrive. In contrast, we consider a *demand-driven* setting in which graph materialization is triggered by observed information demand. Rather than constructing a complete graph before deployment, the system incrementally admits query-relevant source chunks into a persistent semantic graph cache and extracts their entity–relation structure. Previously materialized chunks and their graph structure persist across queries, allowing construction cost to be amortized through reuse. This section first formalizes demand-driven graph construction as a retrieval-agnostic operational paradigm. We then present a pay-as-you-go realization based on top-K retrieval and incremental graph materialization, followed by the amortized objective and construction dynamics used throughout the paper. The admission budget K controls the number of top-ranked source chunks considered for possible materialization per query; it therefore controls the rate at which new semantic structure can enter the persistent graph cache.

Definition 1 (Demand-Driven Knowledge Graph Construction)**.** Given a corpus $\mathcal{D}$ and a query stream $\mathcal{Q}$, demand-driven knowledge graph construction incrementally materializes a persistent graph $\{G_t\}_{t=0}^{T}$, where graph expansion is triggered by observed information demand rather than corpus availability. For each query q_t, a construction policy $\pi : (q_t, G_{t-1}) \to \Delta G_t$ determines the semantic information to materialize. The graph evolves according to $G_t = G_{t-1} \cup \Delta G_t$.

The above formulation separates the operational paradigm from its implementation. The construction policy π may be realized through dense retrieval, uncertainty-aware admission, active exploration, or learned scheduling policies. In this work, we instantiate π using a simple pay-as-you-go admission strategy based on top-K retrieval. This abstraction enables future construction policies to be studied within the same demand-driven framework while preserving a common objective and evaluation.

3.1. Pay-as-you-go Construction Policy

We instantiate the construction policy π using a simple pay-as-you-go strategy that expands the knowledge graph only with semantic information required by the current query. The policy consists of three stages: (i) retrieval of a candidate semantic neighborhood, (ii) selective graph construction over previously unseen evidence, and (iii) graph-based retrieval for downstream question answering. Once semantic information has been materialized, it persists in the graph and is reused by subsequent queries. Let the corpus be partitioned into passages $\mathcal{C} = \{c_1, \ldots, c_N\}$. For each incoming query q_t, a

lightweight retriever (BM25) identifies a candidate semantic neighborhood consisting of the top-K passages, $\mathscr{R}_K(q_t) \subseteq \mathscr{C}$. The system maintains a persistent set of previously materialized passages $\mathscr{M}_t = \mathscr{M}_{t-1} \cup \mathscr{R}_K(q_t), \mathscr{M}_0 = \emptyset$, which records all passages whose semantic information has already been incorporated into the graph. Only passages that have not previously been materialized incur graph construction cost, $\Delta\mathscr{M}_t = \mathscr{R}_K(q_t) \setminus \mathscr{M}_{t-1}$, where $\Delta\mathscr{M}_t$ denotes the newly materialized semantic neighborhood induced by query q_t. For every newly materialized passage $c \in \Delta\mathscr{M}_t$, an entity–relation extractor produces a set of triples, which collectively define the graph update:

$$E(c) = \{(s, r, o)\}, \qquad \Delta G_t = \bigcup_{c \in \Delta\mathscr{M}_t} E(c). \tag{1}$$

The persistent knowledge graph is then updated according to $G_t = G_{t-1} \cup \Delta G_t$, consistent with Definition 1. ΔG_t denotes the corresponding semantic graph expansion. This distinction separates *passage admission*—the selection of previously unseen source chunks from the corpus for persistent storage—from semantic graph construction, which extracts entity–relation triples from admitted chunks. This allows admission policies and extraction strategies to be studied independently.

Finally, graph retrieval is performed over the updated graph. Given query q_t, a graph retriever identifies supporting evidence $R(q_t, G_t)$, which is consumed by a fixed reader $\hat{a}_t = \rho(q_t, R(q_t, G_t))$.

Throughout this work, the admission budget K is the sole tunable parameter governing graph growth: larger K admits a broader semantic neighborhood, increasing construction cost and evidence coverage while defining a different operating point on the quality–cost frontier. Definition 1 characterizes demand-driven graph construction through the policy $\pi : (q_t, G_{t-1}) \to \Delta G_t$, while our implementation realizes this mapping using dense retrieval, selective materialization, and entity–relation extraction: $\mathscr{R}_K(q_t) \xrightarrow{\pi} \Delta G_t$. The materialized passage set $\mathscr{M}_t$ is an implementation-level bookkeeping mechanism that prevents redundant extraction; the evolving graph G_t remains the primary object of interest. This separation allows alternative retrieval, admission, and extraction strategies to be incorporated within the same demand-driven framework and optimization objective.

3.2. Amortized Objective

Demand-driven knowledge graph construction explicitly separates one-time graph construction from recurring graph serving. Construction cost is incurred only when previously unseen semantic information is materialized, whereas serving cost is incurred for every query. Consequently, the objective is not simply to maximize downstream QA performance, but to jointly optimize answer quality, graph construction cost, and serving cost over a persistent query stream.

For a sequence of T queries, the total deployment cost is

$$\mathscr{L}_{\text{Cost}} = \underbrace{\sum_{t=1}^{T} \sum_{c \in \Delta\mathscr{M}_t} \tau_{\text{extract}}(c)}_{\text{Construction}} + \underbrace{\sum_{t=1}^{T} \tau_{\text{read}}(q_t, G_t)}_{\text{Serving}} \tag{2}$$

where τ_{extract} denotes the cost of extracting semantic information from newly materialized passages and τ_{read} denotes the cost of graph retrieval and downstream answer generation.

The corresponding utility is measured by $M_T(K) = \frac{1}{T}\sum_{t=1}^{T} m(\hat{a}_t, a_t)$, where m is Exact Match or token-level F1. Each admission budget K therefore induces an operating point $p_K = (C_T(K), M_T(K))$ on the quality–cost plane. Throughout the remainder of the paper, we characterize the Pareto frontier formed by these operating points to understand when graph construction can be deferred without sacrificing downstream QA quality. The benefit of this amortization is inherently workload dependent: repeated or semantically overlapping queries provide greater opportunity for reuse, whereas low-volume or highly diverse workloads may provide limited opportunity to recover construction costs.

3.3. Formalizing Construction Dynamics

Beyond downstream QA quality, demand-driven graph construction should also be evaluated by how semantic materialization accumulates and is reused over a query workload. Rather than analyzing

individual queries, we characterize the rate of new source-chunk admission over contiguous query blocks, allowing construction and reuse dynamics to be measured at the workload level.

Let the query stream be partitioned into consecutive blocks $\mathscr{B}_1, \ldots, \mathscr{B}_B$ of fixed size. For each block $\mathscr{B}_b$, we compute the average number of newly admitted source chunks, $\bar{\Delta}_b = \frac{1}{|\mathscr{B}_b|} \sum_{q_t \in \mathscr{B}_b} |\Delta \mathscr{M}_t|$, where $|\Delta \mathscr{M}_t|$ denotes the number of previously unseen source chunks admitted at query q_t. As previously admitted chunks are increasingly reused, $\bar{\Delta}_b$ decreases, indicating that progressively less new semantic materialization is required. To summarize this trend, we define the stabilization index $T^*(K) = \min\left\{b : \bar{\Delta}_b < \frac{1}{2}\bar{\Delta}_1\right\}$, which identifies the first evaluation block whose average new-chunk admission falls below half of its initial value. Together, $\{\bar{\Delta}_b\}$ and $T^*(K)$ characterize how quickly the workload transitions from frequent new materialization toward greater reuse under sustained querying. We additionally define the *final cache size* as $|A_T|$, the total number of unique source passages admitted into the persistent graph after processing the complete query workload. This quantity measures the cumulative footprint of demand-driven graph construction (rather than the number of graph nodes, edges, triples, or tokens which are specific to query ordering). quantifying how quickly graph construction transitions from materialization toward reuse under sustained query workloads.

4. Experimental Setup

This section describes the construction policies, datasets, implementation details, and evaluation protocol used to evaluate demand-driven knowledge graph construction.

4.1. Construction Policies Compared

We compare demand-driven graph construction against fully materialized graph baselines and demand-driven non-graph alternatives. All methods share the same retrieval and downstream QA pipeline, differing only in the persistent semantic representation maintained across queries.

- **Full AutoSchemaKG** [6]: Offline schema-aware knowledge graph containing entities, relations, and schema abstractions.
- **Full Entity KG**: Offline entity knowledge graph constructed over the complete corpus [6, 13].
- **Pay-as-you-go Entity KG (ours)**: Demand-driven entity knowledge graph that incrementally materializes entity–relation triples.
- **Lazy Text-RAG**: Demand-driven passage cache without graph construction.
- **Relation-Free Tri-Graph**: Demand-driven entity co-occurrence graph [14] without explicit relation extraction using an LLM.

4.2. Datasets

We evaluate on three widely used multi-hop question answering benchmarks with complementary reasoning characteristics. MuSiQue [3] serves as the primary benchmark throughout the paper due to its explicitly compositional 2-, 3-, and 4-hop questions, making it particularly well suited for evaluating demand-driven graph construction. HotpotQA [4] provides a complementary benchmark containing bridge and comparison questions, while 2WikiMultiHopQA [5] evaluates generalization across different entity distributions and supporting-evidence structures.

4.3. Implementation Details

All construction policies share the same retrieval and downstream QA pipeline to isolate the effect of construction timing. For each query, BM25 retrieves a ranked list of candidate passages, from which the top-K passages are admitted according to the construction policy. Demand-driven graph construction

employs the entity–relation extraction prompt from AutoSchemaKG [6]. Graph retrieval is performed over the currently materialized entity graph, with query-relevant entities from the retrieved evidence used to seed personalized PageRank (PPR). At inference time, Granite-r2 [15] is used to embed the query and seed Personalized PageRank (PPR) over the graph, whose retrieved evidence is provided to Llama4-Maverick [16] for answer generation. Our implementation focuses on entity-level graphs to isolate the effect of construction timing from schema complexity; richer schema-aware graph construction is included only as an offline reference baseline. Unless otherwise stated, prompts, model configurations, and implementation choices remain fixed across all experiments.

4.4. Evaluation Protocol

Demand-driven graph construction is evaluated over persistent query workloads rather than isolated questions, allowing graph construction costs to be amortized through repeated graph reuse. We consider workloads consisting of the first 250 and 1000 queries from each benchmark. The 250-query workload characterizes an early deployment regime, while the 1000-query workload represents a mature operating regime in which substantial portions of the graph have already been materialized.

Queries are processed sequentially using the demand-driven construction policy described in Section 3.1, allowing previously materialized graph regions to be reused by subsequent queries. We evaluate admission budgets $K \in \{1, 2, 5, 10, 20, 40\}$, each corresponding to a distinct operating point on the amortized quality–cost frontier. Each experiment is evaluated over five query-order seeds stratified by question difficulty. Reported results are averaged across seeds. We report Exact Match (EM), token-level F1, supporting-passage Recall@5, graph construction cost, and reader serving cost. Construction cost is charged only for previously unseen passages admitted into the graph, while reader cost is accumulated over the complete workload according to the amortized objective introduced in Section 3.1. Unless otherwise stated, the 1000-query workload serves as the primary evaluation setting throughout the paper, while the 250-query workload serves as a lightweight prototyping regime for studying representative operating points at reduced construction cost.

5. Results

We evaluate demand-driven knowledge graph construction through five research questions that directly correspond to the formulation introduced in Section 3.1. We first investigate whether demand-driven graph construction can recover the performance of fully materialized knowledge graphs (RQ1). We then characterize the resulting quality–cost trade-off under amortized construction (RQ2), analyze how graph construction evolves over sustained query workloads (RQ3), study the contribution of explicit relation-aware graph construction (RQ4), and finally evaluate whether these observations generalize across multiple multi-hop QA benchmarks (RQ5).

5.1. RQ1. Can Demand-Driven Graph Construction Recover the Performance of a Fully Materialized Knowledge Graph?

We first evaluate whether demand-driven graph construction can recover the downstream QA performance of a fully materialized entity knowledge graph under different stages of deployment. The 250-query workload represents an early deployment regime in which graph construction is still actively expanding, whereas the 1000-query workload represents a mature deployment regime where semantic construction has been substantially amortized through graph reuse. Table 2 reveals two consistent trends. First, increasing the admission budget K consistently improves downstream QA performance in both deployment regimes. Materializing progressively larger semantic neighborhoods increases supporting-passage recall, leading to corresponding improvements in EM and F1. Second, the entire quality frontier shifts upward as the workload matures. For every admission budget, the 1000-query workload outperforms its 250-query counterpart, demonstrating that previously materialized graph structure is effectively reused across subsequent queries without modifying either the retriever or the

Table 2
Quality frontier for demand-driven graph construction on MuSiQue under early (250-query) and mature (1000-query) deployment regimes. Increasing the admission budget progressively expands the materialized semantic neighborhood, improving downstream QA performance in both regimes. As the graph matures, the entire quality frontier shifts upward due to accumulated graph reuse.

	250-query workload			1000-query workload		
K	**EM**	**F1**	**R@5**	**EM**	**F1**	**R@5**
1	17.84 ± 1.43	28.88 ± 1.26	27.86 ± 0.26	22.96 ± 0.44	34.51 ± 0.55	29.45 ± 0.36
2	21.52 ± 1.21	32.79 ± 0.98	39.25 ± 0.30	25.80 ± 0.68	38.05 ± 0.97	39.49 ± 0.13
5	24.16 ± 1.28	36.99 ± 1.35	48.28 ± 0.87	27.88 ± 0.78	40.83 ± 0.66	47.19 ± 0.20
10	27.76 ± 1.31	41.82 ± 1.52	54.00 ± 0.46	31.64 ± 1.01	44.58 ± 0.93	53.10 ± 0.56
20	30.24 ± 0.73	43.44 ± 1.26	56.65 ± 0.63	34.32 ± 0.59	47.51 ± 0.83	57.12 ± 0.33
40	**33.52 ± 0.44**	**47.09 ± 1.08**	**60.09 ± 0.67**	**35.02 ± 0.70**	**48.17 ± 0.51**	**58.91 ± 0.40**
Full Entity KG	**39.20**	**54.18**	**67.27**	**36.70**	**51.30**	**62.51**

downstream reader. The highest-performing demand-driven configuration reaches within 3.13 F1 points of the fully materialized entity knowledge graph (see Table 2) while constructing only the semantic neighborhoods required to answer observed queries for the 1000-query workload. More importantly, these results demonstrate that *when* semantic graph construction occurs can be decoupled from *how* the graph is constructed. Rather than requiring complete semantic extraction before deployment, competitive GraphRAG performance can be achieved through demand-driven graph growth that incrementally materializes semantic structure in response to observed information demand.

Takeaway. Demand-driven graph construction exhibits a consistent quality frontier across deployment regimes, with mature workloads benefiting from accumulated graph reuse while progressively approaching the performance of a fully materialized knowledge graph. These findings support our central hypothesis that the timing of semantic graph construction is itself an important systems design choice for GraphRAG.

5.2. RQ2. What Is the Amortized Quality–Cost Trade-off?

Having established that demand-driven graph construction approaches the performance of fully materialized knowledge graphs, we next examine the corresponding quality–cost trade-off. Unlike conventional GraphRAG evaluation, which focuses solely on downstream QA performance, our formulation explicitly accounts for both the one-time cost of semantic graph construction and the recurring cost of serving queries. This enables different construction strategies to be compared under a common amortized quality–cost objective. Tables 2 and 3 characterize this frontier. Increasing the admission budget progressively improves downstream QA performance by materializing larger semantic neighborhoods, but each improvement incurs additional semantic construction. Demand-driven graph construction consistently approaches the performance of a fully materialized entity knowledge graph while avoiding the substantial upfront investment required for complete corpus-wide graph construction. In contrast, fully materializing a schema-aware graph incurs dramatically higher construction cost without corresponding improvements in downstream QA performance.

The results also reveal diminishing returns with increasing admission budgets. While larger values of K continue to improve answer quality, the incremental gains become progressively smaller relative to the additional semantic construction required. This behavior supports the central premise of demand-driven graph construction: semantic extraction should be viewed as an amortized investment whose value is realized through repeated graph reuse rather than as a mandatory offline preprocessing step.

Takeaway. Demand-driven graph construction achieves a favorable amortized quality–cost trade-off, approaching the performance of fully materialized knowledge graphs while substantially reducing the overall cost of semantic graph construction.

Table 3
Representative operating points on the amortized quality–cost frontier for the 1000-query MuSiQue workload. Demand-driven graph construction approaches the performance of a fully materialized entity knowledge graph while substantially reducing overall semantic construction cost. (M = million tokens) *Note.* Construction costs exhibit zero standard deviation because all query-order seeds converge to the same final graph for a fixed admission budget. Reported ±0.00 values indicate nonzero variations that are below the displayed precision.

Method	F1	Construction (M)	Reader (M)	Total (M)
Full AutoSchemaKG	43.25	113.385	3.136	116.521
Full Entity KG	**51.30**	7.264	2.504	9.768
Demand-Driven ($K = 10$)	44.58 ± 0.93	3.187 ± 0.00	2.464 ± 0.001	5.651 ± 0.001
Demand-Driven ($K = 20$)	47.51 ± 0.83	4.925 ± 0.00	2.479 ± 0.004	7.404 ± 0.004
Demand-Driven ($K = 40$)	48.17 ± 0.51	6.451 ± 0.00	2.494 ± 0.004	8.945 ± 0.004

5.3. RQ3. Construction Dynamics Under Sustained Query Workloads

To characterize construction dynamics, we partition the 1000-query workload into five consecutive blocks of 200 queries each and compute the block-level construction statistics introduced in Section 3.3. While RQ2 characterizes the final quality–cost operating point, it does not explain how demand-driven graph construction reaches that operating regime. We therefore analyze the evolution of graph construction over sustained query workloads using the construction dynamics introduced in Section 3.

Table 4 reveals two consistent trends. First, larger admission budgets incur substantially greater semantic construction during the initial stages of deployment by materializing larger semantic neighborhoods. Second, marginal graph construction decreases steadily across all admission budgets as previously materialized graph regions are increasingly reused. This effect becomes progressively stronger with increasing admission budgets, with growth reduction increasing from 25.8% at $K = 1$ to 87.2% at $K = 40$. The stabilization index $T^*(K)$ summarizes this transition from graph construction to graph reuse. Highly selective admission policies ($K \leq 5$) continue admitting new semantic content throughout the observed workload and do not reach the stabilization threshold. In contrast, larger admission budgets ($K \geq 20$) stabilize after only two to three query blocks despite constructing substantially larger semantic neighborhoods, reflected by the greater number of unique source passages admitted into the persistent graph (A_T |) and subsequently materialized into entity–relation triples. These results indicate that greater upfront semantic investment accelerates graph reuse, allowing construction costs to be amortized more rapidly over subsequent queries.

Takeaway. Demand-driven graph construction naturally transitions from an initial construction-intensive phase to a reuse-dominated operating regime. Larger admission budgets require greater upfront semantic investment but amortize semantic construction more rapidly, reaching stable operating regimes earlier through increased graph reuse.

Table 4
Construction dynamics of demand-driven knowledge graph construction on the MuSiQue-1000 workload. Initial and Final Growth report the average number of newly materialized passages per query in the first and final 200-query blocks, respectively. Growth Reduction measures the relative decrease in marginal graph construction over the workload. The stabilization index $T^*(K)$ denotes the first block whose average marginal construction falls below half of its initial value.

K	Initial Growth	Final Growth	Growth Reduction (%)	$T^*(K)$	Final Cache Size $\lvert A_T \rvert$
1	0.94	0.70	25.8	≥ 5	792
2	1.82	1.23	32.5	≥ 5	1482
5	4.32	2.43	43.6	≥ 5	3193
10	8.14	3.70	54.5	4–5	5399
20	14.67	4.35	70.3	3	8189
40	24.71	3.16	87.2	2–3	10321

Table 5
Effect of semantic representation on demand-driven graph construction. All methods employ the same dense retriever, admission policy, and downstream reader; only the persistent semantic representation differs. Results are reported on the 1000-query MuSiQue workload.

Method	K	EM	F1	Recall@5
Lazy Text-RAG	10	15.10 ± 0.60	21.33 ± 0.45	47.96 ± 0.00
	20	15.48 ± 0.37	22.16 ± 0.45	47.92 ± 0.00
	40	15.78 ± 0.26	22.57 ± 0.16	47.91 ± 0.00
Relation-Free Tri-Graph	10	12.06 ± 0.79	18.28 ± 0.92	27.47 ± 0.79
	20	11.62 ± 0.44	17.51 ± 0.66	24.71 ± 0.47
	40	11.02 ± 0.35	17.09 ± 0.27	22.27 ± 0.53
Demand-Driven Entity KG	10	**31.64 ± 1.01**	**44.58 ± 0.93**	**53.10 ± 0.56**
	20	**34.32 ± 0.59**	**47.51 ± 0.83**	**57.12 ± 0.33**
	40	**35.02 ± 0.70**	**48.17 ± 0.51**	**58.91 ± 0.40**

5.4. RQ4. What Semantic Representation Should Be Materialized for Demand-Driven Graph Construction?

Having established the quality–cost benefits and construction dynamics of demand-driven graph construction, we next investigate whether these gains arise from explicitly materializing entity–relation structure or simply from retaining additional semantic context across queries.

Table 5 compares the three demand-driven construction policies introduced in Section 4. All methods employ the same dense retriever, admission policy, and downstream reader, differing only in the persistent semantic representation maintained across queries. Across all admission budgets, the demand-driven entity KG consistently outperforms both Lazy Text-RAG and Relation-Free Tri-Graph. While persistent text caches improve retrieval through evidence reuse, and relation-free graphs capture coarse entity connectivity, neither sufficiently represents the semantic relationships required for multi-hop reasoning. Explicit relation extraction instead preserves the semantics of how entities are connected, enabling graph retrieval to activate reasoning paths that text caches and co-occurrence graphs cannot reliably recover. Consequently, explicit relation extraction consistently improves both supporting-passage recall and downstream QA performance, demonstrating that the benefits of demand-driven graph construction arise from richer semantic representations rather than persistence alone.

Takeaway. Explicit relation extraction is essential for realizing the full benefits of demand-driven graph construction, consistently outperforming both persistent text caches and relation-free graph representations under an otherwise identical retrieval pipeline.

5.5. RQ5. Does the Demand-Driven Paradigm Generalize Across Multi-Hop QA Benchmarks?

Finally, we evaluate whether the demand-driven construction paradigm generalizes beyond MuSiQue. Using the same retrieval pipeline, construction policy, and downstream reader, we repeat the evaluation on HotpotQA and 2WikiMultiHopQA without modifying the underlying methodology. Representative admission budgets ($K = \{10, 20, 40\}$) are compared against fully materialized entity knowledge graphs.

Table 6 shows that the quality–cost frontier generalizes consistently across all three benchmarks. Increasing the admission budget improves downstream QA performance in every dataset, while the marginal gains diminish beyond moderate construction budgets. However, the preferred operating point differs across datasets. On MuSiQue, performance continues to improve through $K = 40$, reflecting its greater compositional reasoning complexity. In contrast, HotpotQA and 2WikiMultiHopQA recover nearly all attainable performance at $K = 20$, with only modest gains from further graph construction. Demand-driven construction is therefore not universally cheaper than full graph materialization; when admission budgets become sufficiently large (e.g., HotpotQA at K=40), its construction cost can approach

Table 6
Generalization of the quality–cost frontier across multi-hop QA benchmarks. Representative admission budgets ($K = 10, 20, 40$) illustrate how downstream QA quality evolves with increasing semantic construction. Full Entity KG denotes complete offline graph construction over the corpus. (M=millions)

Dataset	Method	K	EM	F1	Recall@5	Constr. (M)	Constr. %	ΔF1
MuSiQue	Dem-Driv.	10	31.64±1.01	44.58±0.93	53.10±0.56	3.19	43.9%	-6.72
	Dem.-Driv.	20	34.32±0.59	47.51±0.83	57.12±0.33	4.93	67.8%	-3.79
	Dem.-Driv.	40	35.02±0.70	48.17±0.51	58.91±0.40	6.45	88.8%	-3.13
	Full Entity KG	–	36.70	**51.30**	62.51	7.26	–	–
HotpotQA	Dem.-Driv.	10	57.96±0.53	73.55±0.66	83.29±0.35	8.17	85.7%	-3.40
	Dem.-Driv.	20	60.74±0.63	76.41±0.45	89.05±0.31	9.27	97.3%	-0.53
	Dem.-Driv.	40	61.32±0.45	**77.24**±0.39	90.73±0.32	9.49	99.6%	+0.30
	Full Entity KG	–	61.30	76.94	91.17	9.53	–	–
2WikiMulti	Dem.-Driv.	10	64.44±0.56	72.65±0.55	72.68±0.30	4.62	79.1%	-5.88
	Dem.-Driv.	20	66.56±0.50	74.62±0.44	78.06±0.29	5.38	92.1%	-3.91
	Dem.-Driv.	40	68.12±0.22	76.60±0.22	82.60±0.17	5.69	97.5%	-1.93
	Full Entity KG	–	69.40	**78.53**	87.76	5.84	–	–

the offline baseline, reflecting the cost of incremental online construction rather than a one-time offline build.

These results demonstrate that demand-driven graph construction naturally adapts to workload characteristics rather than requiring a fixed construction policy. More challenging reasoning tasks benefit from larger semantic neighborhoods, whereas simpler workloads achieve comparable performance with substantially lower construction cost.

Takeaway. The demand-driven construction paradigm generalizes across diverse multi-hop QA benchmarks, while the optimal operating point depends on the reasoning characteristics of the workload rather than a universal admission budget.

5.6. Beyond the Quality–Cost Frontier: Reasoning Complexity

Figure 2 stratifies QA performance by reasoning complexity to examine where additional semantic construction is most beneficial. We evaluate up to $K = 40$; $K = 80$ is included for the 250-query workload as an upper bound, but incurs a construction cost comparable to full graph materialization and is therefore impractical.

Increasing K improves performance across reasoning depths, but gains are larger for longer reasoning chains. While 2-hop performance saturates relatively quickly, 3-hop and especially 4-hop performance continue to benefit from increased graph coverage. However, a gap remains for 4-hop questions, indicating that additional construction alone cannot fully recover the most challenging reasoning chains.

These results complement RQ1 and RQ3: larger admission budgets provide greater evidence coverage and subsequent reuse, improving QA quality, but with diminishing returns once sufficient supporting evidence has been materialized. The strongest gains on higher-hop questions suggest that demand-driven construction primarily benefits complex reasoning by increasing access to the bridge evidence required to complete longer chains.

6. Conclusion

We introduced *demand-driven graph construction* as an operational paradigm for GraphRAG, in which semantic structure is materialized in response to user information needs rather than before deployment or as new documents arrive. We formalized this setting through an amortized quality–cost objective and instantiated it using a simple pay-as-you-go policy that incrementally materializes and reuses

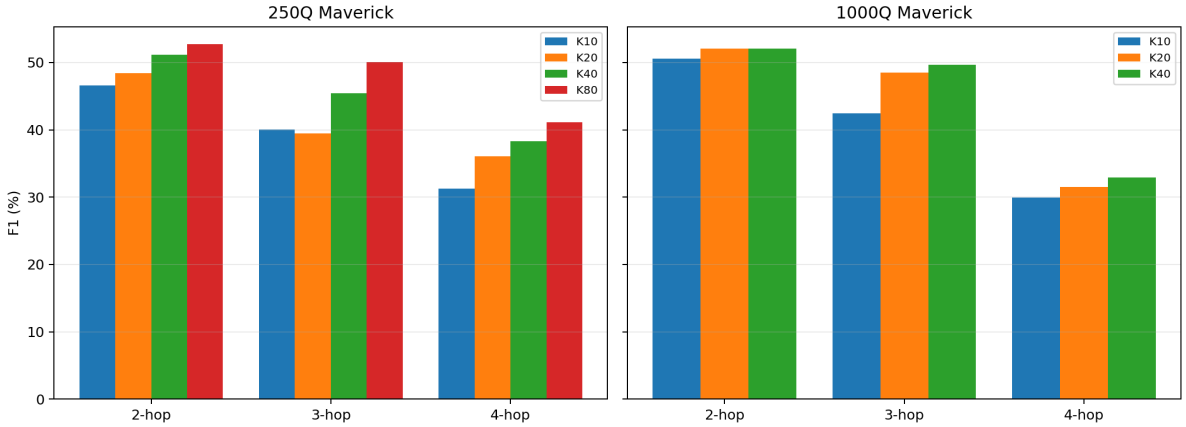


Figure 2: Hop-stratified QA performance on MuSiQue for increasing admission budgets under early (250-query) and mature (1000-query) deployment regimes. The additional $K = 80$ setting illustrates more aggressive graph construction during prototyping (250Q workload).

semantic structure across persistent query workloads. Experiments on MuSiQue, HotpotQA, and 2WikiMultiHopQA show that demand-driven construction can approach the performance of fully materialized knowledge graphs while substantially reducing construction cost, with the preferred operating point depending on workload and reasoning complexity. We further showed that construction transitions naturally from materialization to reuse and that explicit relation extraction is important for realizing the quality–cost benefits of the approach. Our study also highlights important limitations. The benefits of amortization depend on sufficient query volume and semantic overlap; low-volume, highly diverse, privacy-sensitive, or rapidly changing workloads may favor full offline construction. The framework also assumes reliable entity extraction and resolution, while incremental LLM-based extraction can introduce entity drift, alias fragmentation, and consistency challenges. Future work should therefore explore adaptive admission policies, robust entity resolution, and consistency-aware graph maintenance for evolving corpora.

Declaration of use of Generative AI

Generative AI tools were used solely to improve the readability of the manuscript and to correct grammatical and stylistic issues. The authors reviewed all generated suggestions and take full responsibility for the final content.

References

[1] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, J. Larson, From local to global: A graph rag approach to query-focused summarization, arXiv preprint arXiv:2404.16130 (2024).

[2] L. Zhong, J. Wu, Q. Li, H. Peng, X. Wu, A comprehensive survey on automatic knowledge graph construction, ACM Computing Surveys 56 (2023) 1–62.

[3] H. Trivedi, N. Balasubramanian, T. Khot, A. Sabharwal, ♫ musique: Multihop questions via single-hop question composition, Transactions of the Association for Computational Linguistics 10 (2022) 539–554.

[4] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, C. D. Manning, Hotpotqa: A dataset for diverse, explainable multi-hop question answering, in: Proceedings of the 2018 conference on empirical methods in natural language processing, 2018, pp. 2369–2380.

[5] X. Ho, A.-K. D. Nguyen, S. Sugawara, A. Aizawa, Constructing a multi-hop qa dataset for

comprehensive evaluation of reasoning steps, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 6609–6625.

[6] J. Bai, W. Fan, Q. Hu, Q. Zong, C. Li, H. T. Tsang, H. Luo, Y. Yim, H. Huang, X. Zhou, et al., AutoSchemaKG: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora, in: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2026, pp. 20557–20584.

[7] Y. Lairgi, L. Moncla, R. Cazabet, K. Benabdeslem, P. Cléau, iText2KG: Incremental knowledge graphs construction using large language models, in: International Conference on Web Information Systems Engineering, Springer, 2024, pp. 214–229.

[8] C. Bu, G. Chang, Z. Chen, C. Dang, Z. Wu, Y. He, X. Wu, Query-driven multimodal GraphRAG: Dynamic local knowledge graph construction for online reasoning, in: Findings of the Association for Computational Linguistics: ACL 2025, 2025, pp. 21360–21380.

[9] B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, Y. Su, HippoRAG: Neurobiologically inspired long-term memory for large language models, Advances in neural information processing systems 37 (2024) 59532–59569.

[10] E. Curry, Dataspaces: fundamentals, principles, and techniques, in: Real-time Linked Dataspaces: Enabling Data Ecosystems for Intelligent Systems, Springer, 2019, pp. 45–62.

[11] J. F. Sequeda, W. J. Briggs, D. P. Miranker, W. P. Heideman, A pay-as-you-go methodology to design and build enterprise knowledge graphs from relational databases, in: International semantic web conference, Springer, 2019, pp. 526–545.

[12] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, T. Mitchell, Toward an architecture for never-ending language learning, in: Proceedings of the AAAI conference on artificial intelligence, volume 24, 2010, pp. 1306–1313.

[13] Q. Xiao, H. T. Tsang, J. Bai, Terag: Token-efficient graph-based retrieval-augmented generation, arXiv preprint arXiv:2509.18667 (2025).

[14] L. Zhuang, S. Chen, Y. Xiao, H. Zhou, Y. Zhang, H. Chen, Q. Zhang, X. Huang, Linearrag: Linear graph retrieval augmented generation on large-scale corpora, arXiv preprint arXiv:2510.10114 (2025).

[15] P. Awasthy, A. Trivedi, Y. Li, M. Doshi, R. Bhat, V. Kumar, Y. Yang, B. Iyer, A. Daniels, R. Murthy, et al., Granite embedding r2 models, arXiv preprint arXiv:2508.21085 (2025).

[16] A. Adcock, A. Srivastava, A. Dubey, A. Jauhri, A. Pande, A. Pandey, A. Sharma, A. Kadian, A. Kumawat, A. Kelsey, et al., The llama 4 herd: Architecture, training, evaluation, and deployment notes, arXiv preprint arXiv:2601.11659 (2026).