MIN-COST FLOW ROUTING FOR EVIDENCE ASSEMBLY IN LONG MULTIMODAL DOCUMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Answering questions about long multimodal documents requires distributing a fixed evidence budget across relevant facets in text, tables, figures, and slides while avoiding near-duplicates. We present FLOWREADER, which formulates evidence selection as a single minimum-cost flow problem with capacity limits over a multimodal content graph. Spectral decomposition identifies latent aspects of query-relevant content and allocates the budget among them in proportion to their spectral energy. These capacity limits enforce aspect coverage during routing without requiring a language-model planning call. Query-conditioned costs prioritize chains of relevant, mutually consistent evidence. Decomposing the optimal flow produces short evidence chains, which a vision-language model reads in parallel and a reasoner reconciles. On VisDoMBench with Qwen3-VL-32B, FLOWREADER achieves the highest macro accuracy (68.9), surpassing the strongest prior system by 2.7 points, leading on three of five subsets and attaining the highest worst-subset accuracy. It uses a measured 17.5 content nodes per query and maintains its lead at 12.9. Ablation studies with a fixed graph, scorer, reader, and judge show that cost design drives accuracy, capacity limits preserve it while using about three-quarters of the reader tokens required by shortest-path routing without these limits on the same network, and spectral aspects align with LLM-generated sub-questions without a planning call[1].

1 INTRODUCTION

Multimodal documents distribute information across text, tables, figures, captions, and pages, often requiring readers to combine content with its surrounding context. Retrieval-augmented generation (RAG) supplies evidence beyond a language model's immediate context (Lewis et al., 2020). However, conventional text retrieval often overlooks document structure: fixed-length chunks can separate relevant content from the context needed to interpret it (Wang et al., 2025b; Suri et al., 2025; Dong et al., 2024; Ma et al., 2024; Gao et al., 2026). Ranking retrieval units independently compounds this problem because a unit's value can depend on what is retrieved alongside it (Li et al., 2024b; Gutiérrez et al., 2024). A table header, numerical cell, or earlier slide may score poorly in isolation yet be essential for interpreting another unit. We refer to this separation of interdependent content as *fragmented evidence*.

Selecting fragmented evidence under a fixed budget requires coordination. Choosing the highest-scoring units can favour the query's most similar facet, retrieving near-duplicate evidence while omitting lower-scoring units needed for a complete answer. Redundant evidence also consumes the reader's limited context (Liu et al., 2024). Existing methods address different aspects of this problem: multimodal retrievers operate on page images or regions (Faysse et al., 2024; Yu et al., 2025; Suri et al., 2025; Wang et al., 2025a; Xu et al., 2025a); graph-based RAG methods connect evidence units (Edge et al., 2024; Guo et al., 2024; Wan & Yu, 2025; Yan et al., 2025; Du et al., 2026); and joint selection methods use submodular, diversity, or transport objectives (Lin & Bilmes, 2011; Carbonell & Goldstein, 1998; Kulesza & Taskar, 2012; Cuturi, 2013). Recent work also considers coverage and retrieval cost in RAG (Zhang et al., 2026; Wu et al., 2026).

G^2-Reader uses a large language model (LLM) planner to generate sub-questions, retrieve evidence for each, and assess whether the evidence is sufficient (Du et al., 2026), following an iterative re-

[1] Code

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

trieval approach (Trivedi et al., 2023; Asai et al., 2023). Such planning makes evidence needs explicit but does not guarantee an optimal allocation of the retrieval budget: evidence retrieved for different sub-questions may overlap, while sequential planning adds latency.

We introduce FLOWREADER, which represents query decomposition through capacity constraints in a flow network. Spectral decomposition identifies latent aspects of query-relevant document content, each represented by a node with capacity proportional to its spectral energy. A single minimum-cost flow routes a fixed demand across these aspects. Node scores determine entry and termination costs, while edge costs favour transitions to relevant evidence that agrees with preceding nodes. Since the total demand equals the sum of aspect capacities, every feasible solution must satisfy the prescribed allocation. Coverage is therefore a feasibility condition rather than a soft preference (Proposition 1). The optimal flow is decomposed into concise evidence chains, which a vision-language model reads in parallel before a reasoner combines the findings into an answer. Unlike flow-based graph RAG methods that prune relational paths (Chen et al., 2025) or diffuse query mass (Zhou et al., 2026), FLOWREADER explicitly allocates retrieval demand across latent query aspects without an LLM planning call.

On VisDoMBench, FLOWREADER achieves a macro accuracy of 68.9, exceeding the strongest prior system by 2.7 points. It leads on three of five subsets and achieves the highest worst-subset accuracy. The system reads an average of 17.5 distinct content nodes per query and retains a 1.8-point lead when this is reduced to 12.9 nodes. Its retrieval and answering pipeline uses only open-source models, including an open embedder in place of G^2-Reader's proprietary component. With the graph, node scores, reader, and judge held fixed, ablations show that node credit and the agreement objective improve accuracy by 2.3 and 3.4 points, respectively ($p < 0.01$). Capacity constraints reduce reading cost while preserving accuracy: shortest-path routing on the same network without these constraints achieves comparable accuracy but produces chains twice as long. FLOWREADER uses 73% of the reader tokens required by this alternative. Spectral aspects achieve accuracy within 0.45 points of LLM-generated sub-questions ($p = 0.35$), without a planning call.

Our contributions are: **(1)** an exact minimum-cost flow formulation for multimodal evidence selection, with aspect capacity constraints that enforce the prescribed coverage in every feasible solution; **(2)** a spectral construction of aspects and capacities that achieves comparable accuracy to LLM-generated sub-questions without a decomposition call; and **(3)** state-of-the-art accuracy on VisDoMBench, supported by controlled ablations showing that cost design improves accuracy and capacity constraints preserve it while reducing reader token use to 73% of unconstrained routing. We report costs in content nodes, tokens, and model calls.

2 RELATED WORK

Multimodal and graph-structured RAG. Recent systems retrieve evidence directly from page images or multimodal regions. ColPali (Faysse et al., 2024) uses late interaction over visual tokens, while VisRAG (Yu et al., 2025), M3DocVQA (Cho et al., 2025), VisDoMRAG (Suri et al., 2025), and ViDoRAG (Wang et al., 2025a) extend visual retrieval to multi-page and multi-document question answering. RAG-Anything (Guo et al., 2025) integrates text, tables, and images using layout-aware parsing (Wang et al., 2024a). Relevant benchmarks include VisDoMBench (Suri et al., 2025), SlideVQA (Tanaka et al., 2023), SPIQA (Pramanick et al., 2024), and UDA (Hui et al., 2024), which incorporates FeTaQA (Nan et al., 2022) and QASPER (Dasigi et al., 2021).

Graph-based RAG introduces explicit relationships between evidence units. GraphRAG (Edge et al., 2024) and LightRAG (Guo et al., 2024) construct entity–relation graphs, HippoRAG (Gutiérrez et al., 2024) applies personalised PageRank to a knowledge graph, and MMGraphRAG (Wan & Yu, 2025) incorporates multimodal nodes. G^2-Reader (Du et al., 2026) maintains two evolving graphs consolidated through VLM message passing, an approach related to graph memories for long-horizon agents (Chhikara et al., 2025; Xu et al., 2025b; Rasmussen et al., 2025). Flow-based methods use graph structure to prune relational paths (Chen et al., 2025) or diffuse query mass with retrieval guarantees (Zhou et al., 2026).

Adaptive control and optimization-based selection. Iterative retrieval alternates evidence acquisition with reasoning (Press et al., 2023; Yao et al., 2023; Trivedi et al., 2023; Jiang et al., 2023; Shao et al., 2023; Asai et al., 2023). Adaptive methods adjust retrieval depth or query decompo-

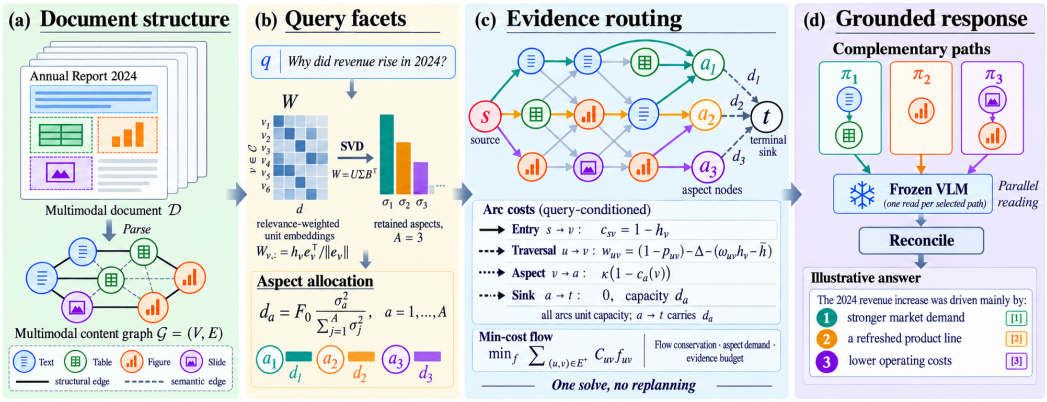


Figure 1: **Overview of FLOWREADER.** *(a)* Each document is parsed once into a multimodal content graph. *(b)* SVD of relevance-weighted node embeddings identifies latent aspects and divides the demand F_0 into aspect capacities d_a. *(c)* A single minimum-cost flow routes this demand from source s through content and aspect nodes to sink t. Aspect capacities enforce the prescribed allocation without a planning call. *(d)* The flow is decomposed into evidence chains, which a VLM reads in parallel before a reasoner combines the findings.

sition to query complexity (Jeong et al., 2024; Yu et al., 2024; Nguyen et al., 2025), while dual-process designs combine fast and reflective stages (Wang et al., 2024b; Cheng et al., 2025; Hu et al., 2026). Liang et al. (2025) survey these approaches. Such methods use model calls to guide retrieval decisions; FLOWREADER determines the allocation in a single optimization before reading any evidence.

Evidence selection has also been formulated using integer linear programming (McDonald, 2007; Gillick & Favre, 2009), submodular maximization (Lin & Bilmes, 2010; 2011), MMR (Carbonell & Goldstein, 1998), and determinantal point processes (Kulesza & Taskar, 2012; Cho et al., 2019). RAG-specific approaches include SMART-RAG (Li et al., 2024a) and demand-coverage objectives (Zhang et al., 2026). Other formulations use optimal transport (Kusner et al., 2015; Cuturi, 2013; Tang et al., 2022; Shaaban et al., 2025) or stochastic passage selection (Zamani & Bendersky, 2024; Meng et al., 2025). Replicator dynamics provide a continuous relaxation of clique selection (Pelillo, 1998; Bomze et al., 2000), motivating our quality–diversity path selector.

Many selection objectives reward coverage or penalise redundancy through terms that compete with other scores. FLOWREADER instead encodes a prescribed allocation across spectral aspects through network capacities and computes an exact minimum-cost flow. When the full demand is routed, every feasible solution satisfies this allocation. This guarantee applies to the complete routed flow over spectral aspects; it does not guarantee semantic coverage or preservation of the allocation after path selection.

3 METHODOLOGY

FLOWREADER assembles evidence through a single minimum-cost flow optimization over a query-conditioned document graph. Given a query q, arc capacities allocate a fixed retrieval demand across latent aspects of relevant content. The optimal flow selects and orders content nodes into evidence chains, which are read in parallel and combined into an answer. Routing and selection are completed before reading, with no replanning or dependencies between reading calls. Implementation details appear in Appendix A.

3.1 MULTIMODAL GRAPH AND QUERY RELEVANCE

We represent the document collection as a graph $G = (V, E)$ whose nodes correspond to passages, tables, figures, and slides. Each node stores its content, summary, modality, and embedding $\mathbf{e}_v$. We adopt the offline graph construction of G^2-Reader (Du et al., 2026), retaining its node-analysis and

linking prompts except for the visual-node prompt, which we modify to include in-image text in the node's keywords. The graph is constructed once per collection and reused across queries. During routing, we add symmetric nearest-neighbor edges; edge types and neighborhood size are specified in Appendix A.1.

For query embedding $\mathbf{q}$, we define the label-aware relevance score and its max-normalized form as

$$r_v = \mathrm{clip}\big(\cos(\mathbf{q}, \mathbf{e}_v) + \rho_{\mathrm{ref}}\, \ell_v(q),\ 0,\ 1\big), \qquad h_v = \frac{r_v}{\max_{u \in V} r_u}, \tag{1}$$

where $\ell_v(q) \in [0, 1]$ measures how closely node v matches a figure or table label explicitly named in the query. Partial matches receive proportionally smaller boosts, and nodes without a matching reference receive none. The parameter ρ_{ref} controls the boost strength. We use h_v throughout routing to determine entry costs, contribute to traversal costs, and construct the spectral aspects and their capacities.

3.2 Query-Conditioned Costs and Aspects

Traversal costs. A useful transition leads to relevant evidence that adds information beyond the preceding node. We assign each edge a utility p_{uv} that combines alignment with the residual query, shared rare terms, and a bonus for transitions between visual and non-visual content (Appendix A.2). Its traversal cost is

$$w_{uv} = (1 - p_{uv}) - \Delta - \big(\omega_{uv} h_v - \overline{h}\big), \tag{2}$$

where $\overline{h} = |V|^{-1} \sum_{v \in V} h_v$ is the mean node score, Δ is a constant offset that sets the cheapest hop cost to zero before node credit, and ω_{uv} measures embedding agreement between the endpoints. The final term provides a *node credit*: it lowers the cost when agreement-weighted destination relevance exceeds the mean node score. A chain of relevant, consistent evidence can therefore cost less than a single passage.

We orient content edges toward strictly higher relevance, so chains progress from supporting context to more relevant nodes. This orientation makes the network acyclic and prevents negative-cost cycles despite the node credit. Most chains contain a single node (75.8%; mean length 1.46; Appendix F).

Nominal flow demand. We derive the nominal demand from the participation ratio of thresholded relevance scores:

$$z_v = \max(0, h_v - \tau), \qquad F_0 = \mathrm{clip}\left(\frac{\big(\sum_{v \in V} z_v\big)^2}{\sum_{v \in V} z_v^2}, F_{\min}, F_{\max}\right). \tag{3}$$

The threshold τ suppresses low-relevance nodes, while $F_{\min}$ and $F_{\max}$ bound the demand. Because h_v is max-normalized, τ is relative to the highest node score. At the reported operating point, the upper bound is active for 99.7% of queries, making F_0 effectively a fixed budget comparable to that of a fixed-k retriever. However, demand measures units of flow rather than distinct content nodes, since a chain may contain several nodes.

Spectral aspects. We derive aspects from query-relevant document content rather than generated sub-questions. Let $\mathcal{C} = \{v \in V : h_v \geq \tau\}$ be the relevant node set. We form a matrix $W \in \mathbb{R}^{|\mathcal{C}| \times d}$ whose rows are relevance-weighted unit embeddings, $W_{v,:} = h_v \mathbf{e}_v^\top / \|\mathbf{e}_v\|$. Its singular value decomposition is $W = U \Sigma B^\top$, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots$ and right singular vectors $\mathbf{b}_a$. These vectors define directions of variation in the relevance-weighted content, with energy given by the squared singular values. We orient each direction toward the normalized query $\hat{\mathbf{q}} = \mathbf{q}/\|\mathbf{q}\|$:

$$\boldsymbol{\phi}_a = \begin{cases} \mathbf{b}_a, & \langle \mathbf{b}_a, \hat{\mathbf{q}} \rangle \geq 0, \\ -\mathbf{b}_a, & \text{otherwise.} \end{cases} \tag{4}$$

Node v aligns with aspect a according to $c_a(v) = \max\big(0, \cos(\mathbf{e}_v, \boldsymbol{\phi}_a)\big)$. We retain enough aspects to meet an energy threshold η, subject to a demand-dependent cap:

$$A_\eta = \min\left\{k : \frac{\sum_{j=1}^{k} \sigma_j^2}{\sum_j \sigma_j^2} \geq \eta\right\}, \qquad A = \min\big(A_\eta, \mathrm{round}(F_0), |\mathcal{C}|\big). \tag{5}$$

The retained spectral energies determine the aspect capacities:

$$d_a = F_0 \frac{\sigma_a^2}{\sum_{j=1}^{A} \sigma_j^2}, \qquad \sum_{a=1}^{A} d_a = F_0. \tag{6}$$

This construction defines both the aspect directions and their demand allocation without an LLM planning call.

3.3 Minimum-Cost Evidence Assembly

We augment the graph with a source s, one aspect node a per retained direction, and a sink t. Every content node can start or end a path. A source-to-content arc (s, v) has cost $1 - h_v$, a content arc (u, v) has cost w_{uv}, and a content-to-aspect arc (v, a) has cost $\kappa(1 - c_a(v))$, where κ controls the penalty for terminal misalignment. These arcs all have unit capacity. Each aspect connects to t through a zero-cost arc with capacity d_a.

Let $G^+ = (V^+, E^+)$ denote this network, with arc costs C_{uv} and capacities U_{uv}. For a feasible routed demand $F \le F_0$, we solve

$$\begin{aligned} f^\star \in \arg\min_{f} \quad & \sum_{(u,v)\in E^+} C_{uv} f_{uv} \\ \text{subject to} \quad & \sum_{v:(u,v)\in E^+} f_{uv} - \sum_{v:(v,u)\in E^+} f_{vu} = b_u, \quad u \in V^+, \\ & 0 \le f_{uv} \le U_{uv}, \qquad\qquad (u,v) \in E^+, \end{aligned} \tag{7}$$

where $b_s = F$, $b_t = -F$, and $b_u = 0$ otherwise. If the network cannot carry F_0, we reduce the demand to $F = \min(F_0, 0.99\, f_{\max})$, where $f_{\max}$ is the network's maximum flow. This fallback is never required on the evaluation queries. The optimization jointly determines path origins, intermediate evidence, and terminal aspects.

When $F = F_0$, the aspect capacities sum to the routed demand, forcing every feasible solution to saturate every aspect-to-sink arc. The formulation therefore enforces the prescribed allocation:

Proposition 1 (Exactness and coverage)**.** *(i) Problem 7 is a linear program over a network polytope and is solvable exactly in polynomial time. Its constraint matrix is totally unimodular, so an integral optimum exists when the capacities and node balances are integral. (ii) G^+ is acyclic, so any feasible flow decomposes into at most $|E^+|$ source-to-sink paths, with no cycles. (iii) In any feasible flow with $F = F_0$, each aspect a receives exactly d_a units through at least $\lceil d_a \rceil$ distinct content nodes.*

These properties follow from standard network-flow results (Ahuja et al., 1993); proofs appear in Appendix B. The coverage guarantee concerns spectral directions in query-relevant content, rather than the semantic facets required to answer the question. Capacities determine *how much* flow each aspect receives, while the terminal cost $\kappa(1 - c_a(v))$ guides *which* nodes supply it. Neither mechanism prevents semantically similar nodes from serving the same aspect. Separately, the node credit rewards relevance and agreement along a chain. When $F < F_0$, aspect capacities still limit concentration, but need not all be saturated.

We decompose the optimal flow into source-to-sink paths and remove the auxiliary nodes to obtain candidate evidence chains $\Pi = \{\pi_1, \ldots, \pi_K\}$, each with an associated flow amount.

3.4 Path Selection and Answer Generation

Candidate paths may share content nodes. We therefore select a compact subset for reading using either routed flow or a quality–diversity criterion. The aspect constraints apply to the complete routed flow; the selected subset is not guaranteed to preserve that allocation.

Greedy selection. We rank candidate paths by their routed flow and retain the top $B \le K$, where B is the path-selection budget. This rule uses the optimization's flow amounts directly, without an additional path-scoring model.

Method	SPIQA	FetaTab	PaperTab	SciGraphQA	SlideVQA	Average
Single-VLM, no retrieval						
GPT-5	55.2 ±0.1	63.9 ±0.3	37.1 ±0.1	64.1 ±0.3	45.1 ±0.1	53.1 ±0.1
Qwen3-VL-32B	29.9 ±0.1	37.4 ±0.4	34.3 ±0.3	23.1 ±0.2	24.9 ±0.2	29.9 ±0.1
Retrieval-augmented generation (Qwen3-VL-32B-Instruct)						
MA-RAG	45.5 ±0.2	27.7 ±0.2	33.4 ±0.5	29.3 ±0.3	29.4 ±0.2	33.1 ±0.1
GraphRAG	62.7 ±0.2	61.4 ±0.2	42.9 ±0.0	<u>65.8</u> ±0.4	21.7 ±0.0	50.9 ±0.1
RAGAnything	67.7 ±1.0	57.8 ±0.2	42.0 ±1.4	41.6 ±2.6	52.2 ±0.5	52.3 ±0.6
ViDoRAG	68.2 ±0.5	58.7 ±0.4	43.7 ±0.2	37.9 ±0.1	<u>71.7</u> ±0.1	56.0 ±0.1
Deepseek-OCR	63.6 ±0.4	70.3 ±0.1	51.6 ±0.2	61.9 ±0.4	65.7 ±0.1	62.6 ±0.1
MMGraphRAG	69.9 ±0.2	<u>72.4</u> ±0.6	56.4 ±0.6	64.1 ±0.3	54.2 ±0.2	63.4 ±0.2
LightRAG	73.9 ±0.0	64.7 ±0.4	51.0 ±0.0	**75.0** ±0.0	29.6 ±0.0	63.7 ±0.1
VisDoMRAG	<u>75.4</u> ±0.0	61.0 ±0.5	56.2 ±0.2	63.4 ±0.1	69.0 ±0.4	65.0 ±0.1
G^2-Reader	73.2 ±0.2	66.9 ±0.1	<u>57.1</u> ±0.2	61.6 ±0.1	**72.3** ±0.0	<u>66.2</u> ±0.1
FlowReader (ours)	**77.7** ±0.3	**74.0** ±0.8	**62.3** ±0.2	61.8 ±0.4	68.8 ±0.9	**68.9** ±0.4

Table 1: **Accuracy (%) on VisDoMBench.** Baseline results are transcribed from Du et al. (2026) and utilize the Qwen3-VL-32B-Instruct backbone, judge, and evaluation prompt. The reported average for LightRAG corresponds to the value in Du et al. (2026) (mean of subset scores: 58.8). We report mean ± standard deviation over three independent runs. **Bold**: best; <u>underline</u>: second best.

Replicator dynamics. Alternatively, we select chains using annealed replicator dynamics with a payoff that rewards relevance and low node overlap (Appendix C). The reported system uses this approach. Greedy selection is sufficient when routed flow already ranks candidates well, while replicator dynamics provides the largest gains when the candidate pool is large and contains redundant chains (Section 4.3).

Answer generation. We follow the worker–reasoner design of G^2-Reader (Du et al., 2026), with evidence units formed by flow routing. A vision-language worker reads each selected chain's text and images and produces a draft answer to the original query, together with supporting evidence. G^2-Reader's workers instead answer planner-generated sub-questions in dependency order, followed by a sufficiency check. In FLOWREADER, all chains are read in a single parallel round.

A reasoner receives the drafts and their evidence, combines complementary findings, and resolves conflicts against the source content (Appendix J). An additional call identifies the question type and expected answer length to set the reasoner's evidence limits; it does not decompose the query. Routing and path selection depend only on the query and document graph, never on the results of earlier reading calls.

4 EXPERIMENTS

4.1 SETUP

Dataset and baselines. We follow the VisDoMBench (Suri et al., 2025) evaluation protocol and use the data released in the public repository of Du et al. (2026).[2] The benchmark comprises five subsets: FetaTab (Wikipedia tables), PaperTab (scientific tables), SPIQA (scientific figures, tables, and text), SciGraphQA (academic graphs), and SlideVQA (slide decks). Each question is paired with five documents containing relevant evidence and distractors. We evaluate all released queries, scoring duplicate questions in the SciGraphQA file once; this leaves the mean unchanged. Baseline results are taken from Du et al. (2026).

Implementation and evaluation. FLOWREADER uses MinerU (Wang et al., 2024a) for document parsing, 768-dimensional nomic-embed-text embeddings (Nussbaum et al., 2024), and Qwen3-VL-32B-Instruct (Bai et al., 2025), served through vLLM at temperature 0. G^2-Reader instead uses 1,536-dimensional text-embedding-3-small embeddings. We score answers using GPT-4o-mini at temperature 0 with the exact evaluation prompt of Du et al. (2026), given in Prompt P7, ensuring that

[2] https://github.com/DorothyDUUU/G2_Reader/tree/main

	Budget		Accuracy (%)					
τ	Chains	Nodes	SPIQA	FetaTab	PaperTab	SciGraphQA	SlideVQA	Avg.
0.5	16.0	17.5	**78.0**	73.3	62.3	**61.7**	**69.4**	**68.9**
0.8	10.9	12.9	76.9	**74.6**	**62.9**	58.0	67.4	68.0
0.9	6.0	8.6	75.5	73.2	58.8	53.1	61.2	64.3

Table 2: **Sensitivity to retrieval demand.** We vary the relative threshold τ while keeping the graph, node scores, reader, reasoner, and judge fixed. Each row evaluates the full dataset, with budgets measured from the chains actually read. The top row uses the reported configuration and corresponds to one of the three runs averaged in Table 1.

our results and the reported baselines use the same rubric. We report means and standard deviations over three independent runs launched on different days. Judge agreement across repeated scoring is 96–99%. Further protocol details appear in Appendix D.

Retrieval budget. Following Du et al. (2026), we express retrieval cost in content nodes and additionally report reader tokens and model calls. FLOWREADER reads 16 chains per query, covering an average of 17.5 distinct nodes and requiring 18 VLM calls. G^2-Reader retrieves up to $k = 5$ nodes per sub-question, with up to six sub-questions, or eight after refinement (Du et al., 2026). These settings give a configured maximum of 30 or 40 nodes, respectively.

Hyperparameters. Following Du et al. (2026), we conduct ablations on the same 250 randomly sampled examples released with G^2-Reader, drawn from the five subsets. We set the demand cap to $F_{\max} = 16$ before any sweep, confirmed this choice on these examples, and held it fixed across all datasets and variants except the demand sensitivity analysis. Table 2 reports this analysis on the full evaluation set: reducing the budget by 26% lowers macro accuracy by 0.9 points. We sweep each of the three remaining constants, η, ρ_{ref}, and b, on the same 250 examples (Table 8).

4.2 MAIN RESULTS

Overall performance. Table 1 reports accuracy on VisDoMBench. FLOWREADER achieves a macro average of 68.9 ± 0.4, outperforming the strongest prior system, G^2-Reader, by 2.7 points. It leads on SPIQA (77.7), FetaTab (74.0), and PaperTab (62.3). The pipeline uses only open-source models, including 768-dimensional nomic-embed-text embeddings, whereas G^2-Reader uses proprietary 1,536-dimensional OpenAI text-embedding-3-small embeddings. Only the evaluation judge, shared across all baselines, is proprietary. Prior systems show uneven performance across subsets. LightRAG leads on SciGraphQA but scores 29.6 on SlideVQA, while G^2-Reader leads on SlideVQA but scores 57.1 on PaperTab. FLOWREADER achieves the highest worst-subset accuracy (61.8), indicating stronger performance on its weakest document type. It also exceeds the same backbone answering directly from pages by 39.0 macro accuracy points.

Subset results and computational cost. FLOWREADER exceeds the best prior score on PaperTab by 5.2 points, SPIQA by 2.3, and FetaTab by 1.6. These subsets often require combining tables or figures with supporting text. The macro standard deviation across runs is 0.4 points, small relative to the 2.7-point average gain. Each query requires 18 VLM calls: 16 parallel chain reads, one reasoner call, and one question-type call. A single SVD and an exact minimum-cost flow allocate retrieval demand across latent query aspects, without planning, refinement, or sufficiency checks.

Performance at smaller budgets. Figure 2 plots accuracy against the measured node budget as flow demand decreases. At 12.9 nodes, FLOWREADER achieves 68.0 accuracy, retaining a 1.8-point lead over G^2-Reader. Interpolation between measured budgets suggests that it matches G^2-Reader's reported accuracy at approximately eleven nodes. At 8.6 nodes, roughly half its default node budget, FLOWREADER achieves 64.3, exceeding all prior systems except VisDoMRAG and G^2-Reader. The default operating point of 17.5 nodes results from the demand setting fixed using the 250-example subset (Table 2).

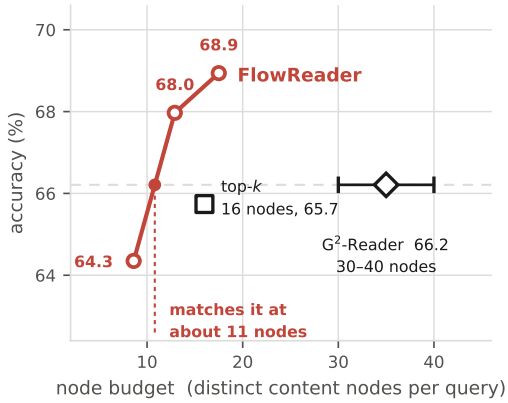


Figure 2: **Accuracy against the measured node budget.** FLOWREADER at three demands meets G^2-Reader's reported accuracy (dashed) at about eleven nodes. G^2-Reader's marker spans its configured budget of 30–40 nodes (Du et al., 2026).

Variant	SPIQA	FetaTab	PaperTab	SciGraphQA	SlideVQA	Avg.
FlowReader	77.6	73.9	62.2	61.5	67.8	68.6
(a) Selection in place of flow, same graph, node scores and reader model						
w/o flow routing (top 16 by relevance)	73.8	73.3	59.5	57.3	64.8	65.7
MMR, relevance–diversity (top 16)	77.9	72.3	60.4	55.6	67.4	66.7
personalised PageRank (top 16)	57.7	59.6	60.8	57.8	64.6	60.1
ranking, same selection stage (50 → 16)	60.5	72.0	44.8	59.3	70.1	61.3
(b) Alternative routing over the same graph and node scores						
min-cost flow	77.6	73.9	62.2	61.5	67.8	68.6
shortest path	76.4	72.6	64.1	63.7	66.7	68.7
k-shortest paths	77.2	71.3	61.8	59.4	67.5	67.5
random walks	76.9	70.4	57.4	57.7	54.6	63.4
PPR chains	26.8	44.3	23.3	19.3	50.6	32.9
(c) Chain selector, min-cost flow routing held fixed						
replicator dynamics, 16 reads fixed	77.2	74.6	63.4	62.0	66.8	68.8
greedy, 16 reads fixed	77.9	74.9	60.2	64.2	68.5	69.1
(d) Components of the flow formulation						
– agreement edge objective	78.7	74.3	62.2	44.2	66.8	65.2
– node credit	72.6	73.6	62.2	57.8	65.6	66.3
– relevance order (document order)	45.1	63.5	40.8	50.1	65.6	53.0
– spectral aspect nodes (single sink)	39.3	54.7	31.4	40.2	58.3	44.8

Table 3: **Ablation of the routing stage** (Section 4.3). Single runs from one batch; the FLOWREADER row is a within-batch re-run of Table 1. Paired tests are in Appendix H: differences of 2.6 points or more are significant ($p \leq 0.001$), those of about one point or less are not ($p \geq 0.19$).

4.3 ABLATIONS

Table 3 varies one mechanism at a time while keeping the graph, node scores, reader, reasoner, and judge fixed. Every variant reads 16 chains.

(a) Alternative evidence selectors. Selecting the 16 highest-scoring nodes without flow routing achieves 65.7, which is 2.9 points below FLOWREADER. Restricting both full-set runs to the 250 examples used in G^2-Reader's ablations, removing flow reduces accuracy from 71.0 to 67.3. This drop is comparable to the decrease from 68.0 to 63.6 reported for G^2-Reader without its planning graph (Du et al., 2026). Other selectors operating on 16 nodes also perform below FLOWREADER: MMR achieves 66.7 ($p = 0.042$), and personalised PageRank achieves 60.1 ($p = 2.5 \times 10^{-19}$).

Applying quality–diversity selection to flow-generated chains preserves accuracy (68.8, compared with 69.1 for greedy selection). Applying the same selection criterion to a ranked list instead yields 61.3, which is 7.3 points below the main FLOWREADER result. The paired comparison favours

FLOWREADER on 292 queries and the ranked-list variant on 127 ($p = 4.7 \times 10^{-16}$), with the largest losses on SPIQA and PaperTab.

(b) Capacity constraints reduce reading cost. We compare flow routing with shortest-path routing on the same network, retaining the costs, edge orientation, and aspect terminals but removing capacity constraints. Accuracy is comparable (68.7 for shortest paths versus 68.6 for flow), with 134 versus 138 discordant queries ($p = 0.86$, McNemar's exact test). This result suggests that the cost design, examined in (d), accounts for the accuracy gains, while capacity constraints primarily reduce reading cost. Without capacities, mean chain length increases from 1.46 to 3.09 nodes, and total node reads per query, including repeated content, increase from 23.3 to 49.4. Reader prompt tokens rise from 46.1k to 63.1k, while both methods use 18 VLM calls (Appendices E and F). By limiting the flow assigned to each aspect, capacity constraints maintain comparable accuracy using 73% of the reader tokens.

(c) Replicator dynamics benefits larger candidate pools. At the reported configuration, routed flow already ranks candidates well: replicator dynamics achieves 68.8, compared with 69.1 for greedy selection at a matched budget. When the candidate pool is doubled, both methods become less accurate, but replicator dynamics performs better than greedy selection (61.6 versus 57.5). It leads on every subset, with a significant paired difference ($p = 1.6 \times 10^{-7}$, McNemar's exact test), despite reading fewer nodes (15.9 versus 16.6). This result supports using a payoff that rewards relevance and low overlap when the candidate pool contains more redundant chains.

(d) Cost components and edge orientation. Removing node credit eliminates the incentive to extend paths: chains collapse to single nodes (16.0 reads over 15.2 distinct nodes), and accuracy falls to 66.3. Removing the agreement term from the edge objective reduces SciGraphQA accuracy from 61.5 to 44.2, suggesting that relevance alone is insufficient to connect compatible graph evidence. Orienting edges by document order rather than relevance changes both the available paths and the order of evidence presented to the reader, reducing accuracy to 53.0.

Replacing the aspect nodes with a single sink yields 44.8, but this variant also changes the cost balance. The derived termination scale κ increases approximately fivefold (median 6.2 versus 1.3), causing termination costs to outweigh relevance-based entry costs. The flow then reaches only 63% of the sixteen highest-scoring nodes, compared with 83% in the full model (Appendix H). This ablation therefore also reflects the effect of calibrating κ in Eq. 10. Chains constructed using personalised PageRank, without the proposed costs or capacities, achieve 32.9.

Spectral aspects and demand allocation. We replace spectral aspects with sub-questions generated by the backbone (Prompt P6) in a paired comparison at a budget of approximately ten nodes. Macro accuracy differs by 0.45 points, with no statistically significant difference ($p = 0.35$). Spectral aspects thus achieve similar observed accuracy without an additional model call. With the termination scale held fixed, allocating demand equally across aspects rather than in proportion to spectral energy reduces accuracy on the 250 ablation examples, in a separate paired run, from 69.2 to 65.5. The largest decline occurs on SciGraphQA (58.0 to 48.0). This trend favours energy-proportional allocation, although the overall difference is not statistically significant ($p = 0.11$).

5 CONCLUSION

Multimodal evidence selection is formulated as a capacitated minimum-cost flow problem, with query decomposition encoded as capacities on spectrally derived aspect nodes. Query-facet coverage becomes a feasibility condition of a single, exactly solvable problem, eliminating the need for language-model-based decomposition. On VisDoMBench, FLOWREADER achieves 68.9 ± 0.4, surpassing the strongest prior system by 2.7 points, attaining the highest worst-subset accuracy, and maintaining a 1.8-point lead at 12.9 content nodes. Controlled ablation studies indicate that costs determine accuracy, while capacities preserve it using only 73% of the reader tokens required by uncapacitated shortest-path routing. Spectral aspects align with LLM-generated sub-questions within 0.45 points, without requiring a planning call. The evaluation is limited to a single benchmark, reader, and judge, with demand and routing fixed prior to evidence reading. Future work includes implementing query-adaptive demand and a second solve conditioned on the initial reading.

Reproducibility Statement

Section 3 describes the method, and Table 4 lists all constants and derived quantities. Appendices A–D provide implementation details, proofs, and the complete evaluation protocol. All prompts appear in Appendix J, including the unmodified evaluation prompt of Du et al. (2026). We use the public VisDoMBench data released through their repository. The pipeline uses open models (MinerU, nomic-embed-text-v1.5, and Qwen3-VL-32B-Instruct), with generation and evaluation performed at temperature 0. Code, configurations, and per-query outputs will be released upon publication.

Ethics Statement

This work uses publicly released benchmark documents and questions and involves no human participants or personal data. Retrieval-augmented systems can produce incorrect answers even when supporting evidence is available. Their outputs therefore require verification before use in high-stakes decisions.

LLM Usage Statement

Large language models support evidence reading, answer generation, and evaluation. Qwen3-VL-32B-Instruct reads evidence and combines findings into answers, while GPT-4o-mini serves as the evaluation judge, following prior work. The authors also used an AI assistant to rephrase sentences and check grammar. The research ideas, experiments, and conclusions are the authors' own, and the authors take full responsibility for the manuscript.

References

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

IR Bomze, Marcello Pelillo, and Volker Stix. Approximating the maximum weight clique using replicator dynamics. *IEEE Transactions on neural networks*, 11(6):1228–1241, 2000.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 335–336, 1998.

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. PathRAG: Pruning graph-based retrieval augmented generation with relational paths. *arXiv preprint arXiv:2502.14902*, 2025.

Rong Cheng, Jinyi Liu, Yan Zheng, Fei Ni, Jiazhen Du, Hangyu Mao, Fuzheng Zhang, Bo Wang, and Jianye Hao. Dualrag: A dual-process approach to integrate reasoning and retrieval for multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31877–31899, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3docvqa: Multi-modal multi-page multi-document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6178–6188, 2025.

Sangwoo Cho, Logan Lebanoff, Hassan Foroosh, and Fei Liu. Improving the similarity measure of determinantal point processes for extractive multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1027–1038, 2019.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, 2021.

Kuicai Dong, Derrick Goh Xin Deik, Yi Quan Lee, Hao Zhang, Xiangyang Li, Cong Zhang, and Yong Liu. Mc-indexing: Effective long document retrieval via multi-view content-aware indexing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2673–2691, 2024.

Yaxin Du, Junru Song, Yifan Zhou, Cheng Wang, Jiahao Gu, Zimeng Chen, Menglan Chen, Wen Yao, Yang Yang, Ying Wen, et al. g^2-Reader: Dual Evolving Graphs for Multimodal Document Comprehension. *arXiv preprint arXiv:2601.22055*, 2026.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models. *arXiv preprint arXiv:2407.01449*, 2024.

Sensen Gao, Shanshan Zhao, Xu Jiang, Lunhao Duan, Yong Xien Chng, Qing-Guo Chen, Weihua Luo, Kaifu Zhang, Jia-Wang Bian, and Mingming Gong. Scaling beyond context: A survey of multimodal retrieval-augmented generation for document understanding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

Dan Gillick and Benoit Favre. A scalable global model for summarization. In *Proceedings of the workshop on integer linear programming for natural language processing*, pp. 10–18, 2009.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2(3), 2024.

Zirui Guo, Xubin Ren, Lingrui Xu, Jiahao Zhang, and Chao Huang. Rag-anything: All-in-one rag framework. *arXiv preprint arXiv:2510.12323*, 2025.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Hao Hu, Yifan Feng, Ruoxue Li, Rundong Xue, Xingliang Hou, Zhiqiang Tian, Yue Gao, and Shaoyi Du. Cog-rag: Cognitive-inspired dual-hypergraph with theme alignment retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31032–31040, 2026.

Yulong Hui, Yao Lu, and Huanchen Zhang. Uda: A benchmark suite for retrieval augmented generation in real-world document analysis. *Advances in Neural Information Processing Systems*, 37: 67200–67217, 2024.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, 2024.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, 2012.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pp. 957–966. PMLR, 2015.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Jiatao Li, Xinyu Hu, and Xiaojun Wan. Smart-rag: Selection using determinantal matrices for augmented retrieval. *arXiv preprint arXiv:2409.13992*, 2024a.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12758–12786, 2024b.

Jintao Liang, Huifeng Lin, You Wu, Rui Zhao, Ziyue Li, et al. Reasoning rag via system 1 or system 2: A survey on reasoning agentic retrieval-augmented generation for industry challenges. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1954–1966, 2025.

Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual conference of the North American chapter of the association for computational linguistics*, pp. 912–920, 2010.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 510–520, 2011.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, et al. MMLongBench-Doc: Benchmarking long-context document understanding with visualizations. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Ryan McDonald. A study of global inference algorithms in multi-document summarization. In *European conference on information retrieval*, pp. 557–564. Springer, 2007.

Siyuan Meng, Junming Liu, Yirong Chen, Song Mao, Pinlong Cai, Guohang Yan, Botian Shi, and Ding Wang. From ranking to selection: A simple but efficient dynamic passage selector for retrieval augmented generation. *arXiv preprint arXiv:2508.09497*, 2025.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, et al. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49, 2022.

Thang Nguyen, Peter Chin, and Yu-Wing Tai. Ma-rag: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *arXiv preprint arXiv:2505.20096*, 2025.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder, 2024.

Marcello Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Advances in Neural Information Processing Systems*, 11, 1998.

Shraman Pramanick, Rama Chellappa, and Subhashini Venugopalan. Spiqa: A dataset for multimodal question answering on scientific papers. *Advances in Neural Information Processing Systems*, 37:118807–118833, 2024.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Mai A Shaaban, Tausifa Jan Saleem, Vijay Ram Kumar Papineni, and Mohammad Yaqub. Motor: Multimodal optimal transport via grounded retrieval in medical visual question answering. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 459–469. Springer, 2025.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9248–9274, 2023.

Manan Suri, Puneet Mathur, Franck Dernoncourt, Kanika Goswami, Ryan A Rossi, and Dinesh Manocha. Visdom: Multi-document qa with visually rich elements using multimodal retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6088–6109, 2025.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. Slidevqa: A dataset for document visual question answering on multiple images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 13636–13645, 2023.

Peggy Tang, Kun Hu, Rui Yan, Lei Zhang, Junbin Gao, and Zhiyong Wang. Otextsum: Extractive text summarisation with optimal transport. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 1128–1141, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Xueyao Wan and Hang Yu. Mmgraphrag: Bridging vision and language with interpretable multimodal knowledge graphs. *arXiv preprint arXiv:2507.20804*, 2025.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, et al. Mineru: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*, 2024a.

Qiuchen Wang, Ruixue Ding, Zehui Chen, Weiqi Wu, Shihang Wang, Pengjun Xie, and Feng Zhao. Vidorag: Visual document retrieval-augmented generation via dynamic iterative reasoning agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9124–9145, 2025a.

Zhitong Wang, Cheng Gao, Chaojun Xiao, Yufei Huang, Shuzheng Si, Kangyang Luo, Yuzhuo Bai, Wenhao Li, Tangjian Duan, Chuancheng Lv, et al. Document segmentation matters for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8063–8075, 2025b.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, et al. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*, 2024b.

Mingyan Wu et al. When knowledge is not free: Cost-aware evidence selection in retrieval-augmented generation. *arXiv preprint arXiv:2606.02245*, 2026.

Mingjun Xu et al. A multi-granularity retrieval framework for visually-rich documents. *arXiv preprint arXiv:2505.01457*, 2025a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025b.

Yuchen Yan et al. Question-adaptive graph learning for multi-hop retrieval augmented generation. *arXiv preprint arXiv:2510.11541*, 2025.

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

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, et al. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tian Yu, Shaolei Zhang, and Yang Feng. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*, 2024.

Hamed Zamani and Michael Bendersky. Stochastic rag: End-to-end retrieval-augmented generation through expected utility maximization. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2641–2646, 2024.

Bingxue Zhang, Jianying Jia, and Feida Zhu. Covering the unseen: Information demand coverage optimization for retrieval-augmented generation. *arXiv preprint arXiv:2606.29328*, 2026.

Zhuoping Zhou, Davoud Ataee Tarzanagh, Sima Didari, Wenjun Hu, Baruch Gutow, Oxana Verkholyak, Masoud Faraki, Heng Hao, Hankyu Moon, and Seungjai Min. Query-aware flow diffusion for graph-based RAG with retrieval guarantees. In *The Fourteenth International Conference on Learning Representations*, 2026.

A Implementation Details

A.1 Content graph

Each document is parsed into text and visual nodes, with one node per text chunk, figure, chart, or table. A node's textual representation is the JSON output of the analysis prompts (Prompt P1 and Prompt P2). We embed this representation using `nomic-embed-text-v1.5`, a 768-dimensional open encoder. Visual nodes use the same encoder to embed the text produced by the visual analysis prompt. Figures and passages are therefore represented in a shared embedding space without a separate image encoder.

The graph contains three edge types, all treated as content edges during routing. *Construction links* are generated by the node-linking prompt (Prompt P3), applied for three iterations when a node is inserted. *Nearest-neighbour edges* connect each node to its k most similar nodes by cosine similarity. These edges are symmetrised, with

$$k = \mathrm{clip}\big(\lceil \log_2 N \rceil,\ 3,\ 32\big), \tag{8}$$

where N is the number of nodes in the document. Inspired by the logarithmic connectivity threshold in random graphs, this rule increases the neighbourhood size with the document size and avoids tuning k separately for each corpus. *Backward edges* reverse construction links when the reverse edge is absent, allowing traversal in either direction before relevance-based orientation. All content edges are then oriented toward higher relevance, as described in Section 3.2.

A.2 Edge score and traversal cost

The edge score p_{uv} in Eq. 2 combines residual query relevance, shared rare entities, and a bonus for transitions between text and visual content:

$$p_{uv} = (1-\beta)\,\mathrm{gain}(u \to v) + \beta\,\mathrm{bridge}(u, v) + \gamma\,\mathbf{1}[\text{modalities differ}], \tag{9}$$

where $\beta = 0.30$ and $\gamma = 0.20$. The residual gain measures how much query-relevant information node v adds beyond node u. A node whose embedding is nearly parallel to that of its predecessor receives little residual gain, discouraging redundant transitions. The bridge term rewards shared rare entities, where an entity is considered rare if it appears in at most $\lceil \sqrt{N} \rceil$ nodes. This threshold is derived from document size. The offset Δ in Eq. 2 shifts the costs so that the cheapest available hop has zero cost before node credit is applied.

A.3 Fixed and derived parameters

We set seven constants and hold them fixed across datasets, models, and runs, except where explicitly varied in ablations. All other quantities in Table 4 are computed from the query or document graph rather than tuned independently.

We derive the termination scale κ by assigning a fraction b of the combined cost variation to the termination term:

$$\kappa = \frac{b}{1-b} \cdot \frac{\mathrm{sd}\big(w_{uv}\big)}{\mathrm{sd}\big(c_a(v)\big)}, \qquad b = 0.634. \tag{10}$$

This calibration scales termination costs relative to traversal-cost variation. For a given variation in aspect alignment, lower variation in traversal costs produces a smaller κ.

The nominal demand F_0 is derived from the query's relevance profile using the participation ratio in Eq. 3. The aspect count then depends on spectral energy and demand, while aspect capacities divide the demand in proportion to the retained spectral energies.

A.4 Models, decoding, and cost

All evidence reading and reasoning use Qwen3-VL-32B-Instruct, the backbone used by the reported baselines, served locally at temperature 0 with a 32,768-token context window. Evidence chains are read in parallel, with one call per chain and at most 6,000 characters of evidence per worker. The reasoner receives each chain's draft answer and up to four supporting nodes, each truncated to 2,000

Quantity	Value
Set by us, fixed across every dataset, model and run	
Reference boost ρ_{ref}	0.30
Cross-modal bonus γ	0.20
Bridge blend β	0.30
Demand floor fraction τ	0.50
Aspect energy η	0.90
Termination balance b	0.634
Demand cap $F_{\max}$, read cap K	16
Derived per query or per document; never tuned	
Neighbour count k	$\text{clip}(\lceil \log_2 N \rceil, 3, 32)$
Bridge rarity cutoff	$\lceil \sqrt{N} \rceil$ nodes
Termination scale κ	Eq. 10, from the spread of w_{uv} and $c_a(v)$
Demand F_0	$\text{clip}(\text{IPR}(z), F_{\min}, F_{\max})$
Aspect count A	energy at η, capped at $\text{round}(F_0)$ and $\lvert\mathcal{C}\rvert$
Aspect capacities d_a	$F_0\, \sigma_a^2 / \sum_{j \le A} \sigma_j^2$
Node score mean $\bar{h}$, relevance mean $\bar{r}$	the query's own means
Entry, traversal, aspect capacities	one unit each

Table 4: **Every quantity the formulation introduces.** Seven constants are ours; the rest are recomputed per query from the query or the document graph. N is the number of nodes in the document and $\mathcal{C} = \{v : h_v \ge \tau\}$ the relevant node set (Section 3.2). Switches that exist in the configuration but are inactive in every reported result (modality boost, diffusion, multi-round routing) are not counted.

characters. This bounds the evidence supplied per chain regardless of its length. Evaluation uses GPT-4o-mini at temperature 0 with a maximum output of 300 tokens, matching the baselines' judge settings and prompt (Prompt P7).

The spectral allocation is concentrated: the leading direction receives a median of 73.7% of the demand, with the remainder distributed across a median of 13 additional directions (Eq. 6). If the network's maximum flow $f_{\max}$ is below F_0, we reduce the routed demand to $F = \min(F_0, 0.99\, f_{\max})$ to ensure feasibility. This can occur when too few content nodes can reach the aspect nodes, but does not occur on any evaluation query.

Graph construction is performed offline once per document. All controlled variants use the same graph, so their comparisons isolate changes made after graph construction.

B PROOFS

Proposition 1(i). The equality constraints in Eq. 7 are defined by the node–arc incidence matrix of G^+. This matrix is totally unimodular, and adding the identity rows associated with the bounds $0 \le f \le U$ preserves this property. Hence, when b and U are integral, every vertex of the feasible polytope is integral. For a feasible instance, the finite capacity bounds make the polytope bounded, so the linear objective attains its optimum at a vertex. An integral optimum therefore exists. Minimum-cost flow admits exact polynomial-time algorithms (Ahuja et al., 1993). Fractional aspect capacities do not affect linearity or the conclusions of parts (ii) and (iii), although an integral optimum is no longer guaranteed.

Proposition 1(ii). Content arcs follow a strict total order defined by relevance, with ties broken by node index. Let $\text{rank}(v) \in \{1, \ldots, |V|\}$ denote a content node's position in this order. Assign potential 0 to s, $1 + \text{rank}(v)$ to each content node, $|V| + 2$ to each aspect node, and $|V| + 3$ to t. Every arc strictly increases this potential, so G^+ is acyclic.

By the flow decomposition theorem (Ahuja et al., 1993), any feasible flow can therefore be decomposed into source-to-sink path flows without cycle flows. Each decomposition step subtracts the minimum positive flow along a path, reducing the residual flow on at least one arc to zero. Thus, at most $|E^+|$ paths are required.

Proposition 1(iii). Only aspect nodes connect directly to t, so all routed flow passes through aspect-to-sink arcs. When $F = F_0 = \sum_a d_a$, flow conservation gives

$$\sum_a f_{at} = F_0 = \sum_a d_a.$$

Since $f_{at} \le d_a$ for every aspect, this equality requires $f_{at} = d_a$ for all a. Conservation at each aspect node then gives $\sum_v f_{va} = d_a$. Each content-to-aspect arc has unit capacity, so carrying d_a units requires at least $\lceil d_a \rceil$ arcs with positive flow. These arcs originate from distinct content nodes, proving the claim. □

C Path Selection by Replicator Dynamics

We treat each candidate chain $\pi_1, \ldots, \pi_K$ as a set of content nodes and define its quality, pairwise overlap, and selection payoff as

$$Q_i = \Big(\frac{1}{|\pi_i|}\sum_{v\in\pi_i} h_v^2\Big)^{1/2}, \qquad R_{ij} = \frac{|\pi_i \cap \pi_j|}{|\pi_i \cup \pi_j|}, \qquad M_{ij} = \begin{cases} Q_i^2, & i = j, \\ \sqrt{Q_i Q_j}\,(1 - R_{ij}), & i \ne j. \end{cases} \tag{11}$$

The diagonal terms reward individual chain quality, while the off-diagonal terms favour relevant chains with few shared nodes. For node-disjoint chains, $R_{ij} = 0$, so the pairwise payoff depends only on quality. The Jaccard term captures node overlap, not semantic redundancy.

Starting from a uniform distribution over candidates, $x_i^{(0)} = 1/K$, replicator dynamics updates their shares as

$$x_i^{(t+1)} = x_i^{(t)} \frac{(Mx^{(t)})_i}{x^{(t)\top} M x^{(t)}}. \tag{12}$$

For the symmetric, nonnegative payoff matrix M, the standard update monotonically increases, or leaves unchanged, the average payoff $x^\top M x$ whenever the denominator is positive. This provides a continuous optimization approach to selecting compatible, high-quality chains (Pelillo, 1998; Bomze et al., 2000).

We anneal the dynamics to sharpen selection progressively. Candidates whose shares fall below a fixed threshold relative to the uniform share $1/K$ are removed, and the remaining distribution is renormalised. Surviving chains are read in descending order of their final shares, up to the read cap. Greedy selection instead ranks chains by their routed flow. The two methods achieve similar accuracy at the reported configuration, while replicator dynamics leads by 4.1 points when the candidate pool is doubled (Section 4.3).

D Evaluation Protocol

Judge and runs. All reported answer accuracies are scored by GPT-4o-mini at temperature 0 using the evaluation prompt of Du et al. (2026), reproduced without modification in Prompt P7. The judge agrees with its own scores on 96–99% of identical predictions evaluated repeatedly. The three runs in Table 1 were launched on different days, with serving infrastructure restarted between runs, and yield a sample standard deviation of 0.35 macro accuracy points. Given the subset sizes, the estimated binomial standard error of macro accuracy is approximately one point.

Ablation samples. Following Du et al. (2026), we use the same 250 randomly sampled examples released with G^2-Reader, drawn from the five subsets. Evaluating each reported configuration on the remaining examples changes its score by at most 0.44 points and preserves all rankings in Tables 1 and 3. The demand cap was fixed before any hyperparameter sweep.

Prompts. We refined the prompts using the ablation examples and failure analysis. One illustrative example in the worker prompt (Prompt P4) includes entities that also appear in three FetaTab evaluation questions. Any direct effect confined to these questions is bounded by one accuracy point on FetaTab and 0.2 points on the macro average.

E READER COST ACCOUNTING

Table 5 reports the cost of reading for each router at the 16-chain budget, measured from the runs of Table 3. Every variant makes the same number of model calls, 16 parallel worker calls plus a reasoner and a query classifier, so the routers differ only in how much evidence each call carries. Wall-clock time was measured on a shared cluster and indicates an order of magnitude; tokens and calls are exact.

Router	Nodes	Prompt tok.	Compl. tok.	Calls	s/query
Min-cost flow (FLOWREADER)	17.5	46,138	1,270	18.0	25.4
k-shortest paths	11.8	50,576	1,697	18.0	34.4
Shortest path	16.9	63,106	2,002	18.0	89.9

Table 5: **Cost of reading per query.** Prompt and completion tokens are summed over every model call of a query. Among routers that form chains, min-cost flow spends the fewest tokens and seconds per query.

F CHAIN STRUCTURE

Table 6 reports the length distribution of the chains actually read. Under min-cost flow, most chains are single nodes and the remainder form a tail up to six nodes and beyond; multi-node chains nonetheless reach 88.5% of queries, so structure is used where the content supports it. Shortest path, which is not bounded by aspect capacities, builds long chains on nearly every query and spends 90.3% of its reads inside them, which is the source of its higher token cost in Table 5.

Router	Chains	Mean length	Single-node	Queries with multi-node
Min-cost flow (FLOWREADER)	35,286	1.46	75.8%	88.5%
k-shortest paths	35,099	1.65	47.5%	94.7%
Shortest path	35,044	3.09	30.0%	96.0%

Table 6: **Chain length distribution** over the full evaluation sets of Du et al. (2026), with at most 16 chains read per query. For min-cost flow the chain counts by length are 26,760 (one node), 4,205 (two), 2,429 (three), 1,048 (four), 490 (five), 214 (six) and 140 (seven or more).

G REFUSAL RATE ACROSS SELECTORS

Figure 3 relates, for each selector and subset, the share of queries on which every worker refused to accuracy. The refusal rate measures only whether the selector put answerable content in front of the readers, independently of reading and judging, and it predicts accuracy across selectors.

H ABLATION DIAGNOSTICS

Each cell of Table 3 is a single run, but every variant answers the same queries as the reported configuration, so each row is tested against it by McNemar's exact test on the paired judgements; for scale, three independent runs of the reported configuration differ by 0.35 macro points. Table 7 also reports what each variant reads. *Reached* is the share of each query's sixteen most relevant nodes, by the relevance of Eq. 1, that a variant reads, and *all refused* the share of queries on which every worker returned no answer. Accuracies are pooled over queries and therefore differ slightly from the macro averages of Table 3.

The single-sink variant reads more distinct nodes than the reported configuration (18.7 against 17.5) yet reaches markedly less of the most relevant evidence, and every worker refuses on 2.6 times as many queries. Within the variant, accuracy follows the evidence reached: it is 19.7% on queries where it reaches under a quarter of the sixteen most relevant nodes, 42.1% between a quarter and a

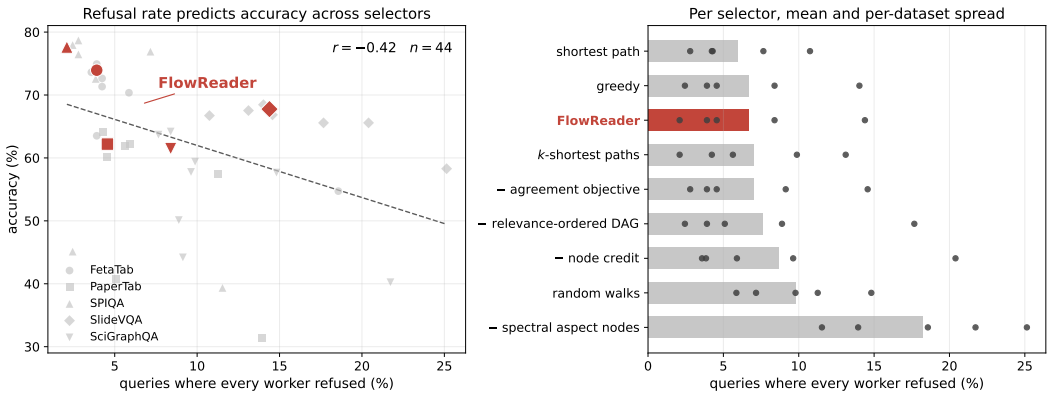


Figure 3: **Refusal rate predicts accuracy.** *Left:* each point is one selector on one subset; the share of queries on which every worker refused predicts accuracy ($r = -0.42$, $n = 44$). *Right:* per selector, the mean and per-subset spread. Removing the spectral aspect nodes raises the all-refused rate 2.6-fold, from 7.0% to 18.2%, far more than any other change.

Variant	Δ acc.	McNemar p	Reached	All refused
Min-cost flow (reported)	—	—	83.2%	7.0%
Shortest path	−0.2	0.86	73.9%	6.2%
k-shortest paths	−1.0	0.19	63.6%	7.2%
Replicator dynamics (16 reads)	0.0	1.00	83.2%	7.0%
Greedy (16 reads)	+0.3	0.68	83.6%	7.0%
− agreement objective	−3.1	1×10^{-4}	80.8%	7.3%
− node credit	−2.6	0.001	81.0%	9.3%
− relevance order	−16.1	$< 10^{-50}$	81.7%	8.1%
− aspect nodes (single sink)	−24.1	$< 10^{-50}$	63.3%	18.2%

Table 7: **Paired significance and evidence reached** for each variant of Table 3, against the reported configuration. *All refused* is pooled over queries. Replicator dynamics with 16 reads fixed is judged differently from the reported configuration on only 115 queries of the evaluation sets of Du et al. (2026), which split 58 to 57, so pooled over queries the two coincide although their macro averages in Table 3 differ by 0.2 points.

half, and 50.1% between a half and three quarters. The calibrated termination scale is what keeps the flow on that evidence. The document-order variant reaches nearly as much of it as the reported configuration with a similar refusal rate, so its loss arises after reading rather than in the evidence selected.

I Sensitivity Analysis

Retrieval demand

Table 2 varies the relative relevance threshold τ in Eq. 3, which determines which nodes contribute to the effective count used to derive demand. All other settings remain fixed. Each row evaluates the full dataset, and node budgets are measured from the chains actually read.

Reducing the node budget by 26% lowers macro accuracy by 0.9 points. Increasing the budget from 12.9 to 17.5 nodes improves SciGraphQA by 3.7 points and SlideVQA by 2.0, while FetaTab and PaperTab change by at most 1.3 points. These results support the reported operating point of 17.5 nodes, the smallest tested budget that avoids a loss on any subset. Below 12.9 nodes, accuracy declines on every subset.

REMAINING CONSTANTS

We vary three constants individually around their reported values: the spectral energy threshold $\eta = 0.90$, the reference boost $\rho_{\text{ref}} = 0.30$, and the termination balance $b = 0.634$. All sweeps use the same 250 ablation examples. A tenth configuration repeats the reported settings on these examples to provide a matched comparison (Table 8). These experiments assess local sensitivity without changing the results in Table 1.

At this sample size, no tested setting is statistically distinguishable from the reported configuration. Macro accuracy varies by 1.5 points as η ranges from 0.70 to 0.95, and by 1.3 points as ρ_{ref} ranges from 0 to 0.45. The termination balance has the largest observed effect. The reported value is the best of the four tested values, with alternatives reducing accuracy by 2.3–3.1 points. This parameter is one of the seven fixed constants; Eq. 10 uses it to derive κ for each query.

The reported configuration scores 69.9 in this separate sweep, compared with 71.0 when the full-set runs are restricted to the same examples (Section 4.3). The difference lies within the ± 5.7-point uncertainty interval at this sample size. We therefore compare configurations within the same run set.

The demand and aspect-count caps are active at different rates. The demand in Eq. 3 reaches $F_{\max}$ for nearly every query, whereas the aspect-count cap $A \leq \text{round}(F_0)$ in Eq. 5 is active for approximately 40% of queries. This cap limits the effect of increasing η, while lowering η can still reduce the number of retained aspects.

Constant	Value	Macro (%)	Δ
Energy threshold η	0.70	69.9	0.0
	0.80	70.7	+0.8
	0.90 (reported)	**69.9**	—
	0.95	69.2	−0.7
Reference boost ρ_{ref}	0	71.1	+1.2
	0.15	69.8	−0.1
	0.30 (reported)	**69.9**	—
	0.45	70.8	+0.9
Termination balance b	0.45	66.8	−3.1
	0.55	69.5	−0.4
	0.634 (reported)	**69.9**	—
	0.75	67.6	−2.3

Table 8: **Sensitivity of the three free constants.** Each is swept one at a time on the 250 ablation samples of Du et al. (2026), with the reported configuration re-run on the same samples so every comparison is on the same samples. Macro is the unweighted mean over the five subsets. A 95% interval on any of these estimates is ± 5.7 points at this sample size, so every Δ in the table is well inside it and no setting is separable from the reported one.

J PROMPTS

Every prompt on a live call path is reproduced below, verbatim from the file the code reads. Placeholders in `$NAME$` form are substituted at call time; the source of each box is printed in its title bar so that any of them can be checked against the repository. Prompts that only an ablation uses are marked as such, and prompts that the repository defines but no call site reaches are not reproduced.

GRAPH CONSTRUCTION

Called once per document, offline, before any query is seen.

P1 Text node analysis Issued once per text chunk while the content graph is built. Its JSON output becomes the node's summary, keywords and tags, which are what the node embedding is computed from.

P1 Text node analysis config/prompts/text_analysis_keyword.txt

```
Your task is to extract ALL concrete entities that explicitly appear in the provided text.

STRICT EXTRACTION RULES:
1. You MUST extract *every* entity that appears in the text.
   - Do not skip or ignore any entity, even if it seems minor.
   - Coverage must be exhaustive.

2. Only extract entities that appear *verbatim* or as clear noun phrases in the text:
   - Named entities (people, organizations, locations)
   - Technical terms and domain-specific terminology
   - Products, models, components
   - Proper nouns and unique identifiers
   - Explicit noun phrases directly present in the text

3. You MUST NOT:
   - Invent new terms
   - Generalize, abstract, or summarize
   - Produce synonyms or inferred concepts
   - Omit any entity actually present
   - Extract raw mathematical formulas, LaTeX expressions, or equations (e.g., $x_{i}$,
  ↪ \alpha, etc.)
   - Include special characters like $, \, {, } in keywords

4. The keywords MUST be entities present in the text, not themes.

5. Preserve the original wording of the entities from the text, but convert mathematical
  ↪ notation to plain text descriptions (e.g., variable x_i instead of $x_{i}$).

6. When the provided content contains no meaningful information, such as reference lists,
  ↪ \
the summary you generate must include exactly No meaningful information to aid filtering.

OUTPUT FORMAT (JSON):
{
  keywords: [
    // ALL entities that appear in the text, in their exact textual form
  ],
  summary: One concise sentence summarizing the content.,
  tags: [
    // Broad classification tags, not entities
  ]
}

------------------------------------------------------------

FEW-SHOT EXAMPLES
------------------------------------------------------------

Example 1
Input Text:
The Tesla Model S uses a Dual Motor system and features an Autopilot module.
It integrates an NVIDIA Drive PX processor and a Panasonic 18650 battery pack.

Output JSON:
{
  keywords: [
    Tesla Model S,
    Dual Motor system,
    Autopilot module,
    NVIDIA Drive PX processor,
    Panasonic 18650 battery pack
  ],
  summary: The text describes components and hardware modules used in the Tesla Model S.,
  tags: [Technology, Automotive, Specifications]
}

Example 2
Input Text:
The DeepMind AlphaFold system achieved state-of-the-art results in the CASP13 competition.
The project also used the UniProt database and the PDB structure repository for training.

Output JSON:
{
  keywords: [
    DeepMind AlphaFold system,
    CASP13 competition,
```

```
    UniProt database,
    PDB structure repository
  ],
  summary: The text describes AlphaFold's performance in CASP13 and the datasets used for
  ↪ its training.,
  tags: [Machine Learning, Biology, Research]
}
```

P2 Visual node analysis Issued once per figure, chart or table image, with the surrounding text and the caption substituted for `$context$` and `$caption$`. It is what makes a visual node comparable to a text node in the same embedding space.

P2 Visual node analysis `config/prompts/image_analysis_ocrkeyword.txt`

```
Generate a structured analysis of the visual elements in the provided image (which is a
  ↪ page or figure from a scientific paper).
You are also provided with the image's surrounding text context (before and after the
  ↪ image) and its caption.

Instructions:
1) Visual Focus: Use the provided context only to aid your understanding of what the
  ↪ image conveys. The summary should be primarily based on the visual evidence and the
  ↪ caption.
2) Keywords: Extract exact in-image terms: labels, legends, axis titles, category names,
  ↪ and domain-specific terms observed in the image. Include any readable text from the
  ↪ image (axis numbers, annotations, data values) as keywords for search indexing.
3) Summary: When a caption is present and includes an index (e.g., Figure 1, Table 2),
  ↪ begin the summary by formalizing it (e.g., Figure X --- ...). Describe only what is
  ↪ visually present and what the caption explicitly states.
4) Tags: Broad categories/themes for classification. At least three tags.
5) Text Content: For the text_content field, you must COMBINE the full caption text AND a
  ↪ transcription of all readable text found inside the image (axis numbers, labels,
  ↪ annotations) into a single raw string. This is for search indexing.

Format the response as a JSON object:
{
    keywords: [
        // Include exact in-image labels, legends, axis titles, category names.
        // Also include readable text from the image (annotations, data values).
        // Order from most to least important.
        // At least three keywords.
    ],
    summary:
        // If caption provides an index, start with Figure X --- ... or Table Y --- ....
        // Concisely describe the visual elements and trends.
    ,
    tags: [
        // Broad categories/themes for classification (e.g., Bar Chart, Ablation Study).
        // At least three tags.
    ],
    text_content:
        // Combine the full caption text AND all readable text from the image into one
  ↪ string.
        // This is used for search indexing.

}

// Few-shot Examples:

Example 1: Map with labeled points
Provided:
- Context: Describes site selection background.
- Caption: Figure 1. Distribution of information centers and viewpoints across the region.
- Visible elements: Map showing markers labeled Information Center and Viewpoint, scale
  ↪ bar 10km.
Expected JSON:
{
  keywords: [Figure 1, Information Center, Viewpoint, Distribution Map, Geospatial,
  ↪ Region, 10km],
  summary: Figure 1 --- Distribution of information centers and viewpoints across the
  ↪ region. The figure presents a regional map with labeled markers indicating locations
  ↪ of information centers and viewpoints, highlighting spatial clustering.,
  tags: [Geospatial, Map, Urban Planning],
  text_content: Figure 1. Distribution of information centers and viewpoints across the
  ↪ region. Information Center Viewpoint 10km
```

```
}

Example 2: Table ablation study
Provided:
- Context: Discusses model variants.
- Caption: Table 2. Ablation study of model components.
- Visible elements: Table columns Method, Accuracy, F1-Score. Rows: Baseline (85.2), Ours
  ↪ (88.9).
Expected JSON:
{
  keywords: [Table 2, Ablation Study, Accuracy, F1-Score, Baseline, Method, 85.2, 88.9],
  summary: Table 2 --- Ablation study of model components. The table compares the
  ↪ Baseline method against the proposed method ('Ours'), showing improvements in
  ↪ Accuracy and F1-Score.,
  tags: [Experiment, Table, Performance Metrics],
  text_content: Table 2. Ablation study of model components. Method Accuracy F1-Score
  ↪ Baseline 85.2 Ours 88.9
}

This is context: {context} and caption: {caption}
```

P3 Node linking Issued when a new node is inserted, against its nearest existing neighbours. The links it returns are the edges of the content graph the flow is later routed over. Run for three iterations throughout.

P3 Node linking `config/prompts/memory_evolve.txt`

```
You are an AI memory evolution agent responsible for managing and evolving a knowledge
  ↪ base.
Here is a memory note (including its content, summary, and keywords), and a few
  ↪ neighboring notes.
Make decisions about its evolution.

The memory note content: {content}
Summary: {context}
Keywords: {keywords}

The {neighbor_number} neighboring notes:
{neighbors}

Based on this information, determine:
1. Which of the neighboring notes should be linked to this memory note?
2. Should the summary and keywords of this memory note be updated, considering its
  ↪ relationships with the neighboring notes?
3. If so, what should be the new summary and keywords?
If you determine that the memory note needs not be updated, the new summary and keywords
  ↪ should be the same as the original summary and keywords.
Please keep the summary concise, informative and to the point, with no more than 30 words.

**Definition of Valid Edge Relationships:**
Two memory notes should be connected ONLY if they exhibit one or more of the following
  ↪ specific logical relationships:

1. **Direct Reference/Citation**: One note explicitly references, cites, or builds upon
  ↪ concepts from another note.
2. **Causal Relationship**: One note describes a cause and another describes its effect,
  ↪ or vice versa.
3. **Part-Whole Relationship**: One note describes a component/part and another describes
  ↪ the whole system, or they describe different parts of the same whole.
4. **Conceptual Elaboration**: One note provides detailed explanation or elaboration of a
  ↪ concept introduced in another note.
5. **Temporal Sequence**: Notes describe sequential steps in a process, chronological
  ↪ events, or stages in a workflow.
6. **Contrastive/Comparative**: Notes discuss the same topic from different perspectives,
  ↪ compare alternatives, or contrast different approaches.
7. **Hierarchical Relationship**: One note describes a general concept and another
  ↪ describes a specific instance or subcategory.
8. **Contextual Dependency**: Understanding one note significantly enhances or requires
  ↪ understanding of another note.

**Exclusion Criteria - DO NOT connect notes if they only:**
- Share common keywords or terminology without substantive conceptual overlap
- Belong to the same broad domain but discuss unrelated specific topics
- Have similar surface-level similarity but no meaningful logical relationship
- Are merely adjacent in the document structure without semantic connection
```

```
- Share only generic concepts (e.g., both mention data or analysis without deeper
  ↪ connection)

**Connection Quality Assessment:**
When evaluating connections, ask yourself:
- Would retrieving one note significantly help understand or contextualize the other?
- Is there a specific, identifiable logical relationship beyond mere topic similarity?
- Would an expert in the domain recognize this as a meaningful connection?
- Could this connection improve retrieval accuracy for relevant queries?
- Are the suggested connections ordered according to relevancy? The most relevant
  ↪ connection should be the first one.

**Summary and Keywords Update Principles:**
When updating the summary and keywords, your goal is to make the memory note more precise
  ↪ and distinctive:
1. **Use neighbors as reference, not as content**: Compare the current note with its
  ↪ connected neighbors to identify what makes this note unique. However, the summary
  ↪ itself should ONLY describe the content of the current note. Do NOT include
  ↪ comparative language such as unlike other notes, different from neighbors, this note
  ↪ does not cover X, or any explicit references to other notes.
2. **Focus on unique aspects**: If the current summary is too generic and overlaps with
  ↪ neighbor summaries, refine it by emphasizing the specific, unique aspects that are
  ↪ present in THIS note's content.
3. **Distinctive keywords**: Update keywords to include terms that uniquely identify this
  ↪ note's specific content. Prefer specific, distinctive keywords over broad, generic
  ↪ ones that many notes might share.
4. **Self-contained description**: The summary should be a self-contained description of
  ↪ the current note's content. It should be clear and informative on its own, without
  ↪ requiring knowledge of other notes to understand it.
5. **When NOT to update**: If the current summary already accurately captures the unique
  ↪ aspects of this note and clearly distinguishes it from neighbors (through its
  ↪ specificity, not through comparative language), do not update it.

**Summary Writing Guidelines:**
- ✓ DO: This note discusses the implementation details of algorithm X, focusing on its
  ↪ time complexity analysis.
- ✓ DO: This note describes the experimental setup for testing hypothesis Y, including
  ↪ specific parameters and measurement protocols.
- × DON'T: Unlike other notes, this note focuses on algorithm X rather than algorithm Y.
- × DON'T: This note does not cover theoretical foundations, which are discussed in
  ↪ other notes.
- × DON'T: This note is different from its neighbors in that it analyzes...

Return your decision in JSON format with the following structure:
{
    suggested_connections: [neighbor_memory_ids],
    should_update: True or False,
    new_summary: new summary,
    new_keywords: [keyword_1,...keyword_n]
}
```

Evidence reading and answer synthesis

Called per query, after the flow has been routed and the chains selected.

P4 Chain reader (worker) Issued once per selected evidence chain, in parallel. `$DOC$` receives that chain's nodes in routed order and `$Q$` the query. This is the only prompt that sees document content at inference time.

P4 Chain reader (worker) `config/prompts/worker_noabstain.txt`

```
Read the evidence passages and images below, then answer the question.

<text>
$DOC$
</text>

Question: $Q$

ANSWERING POLICY (read this first):
Your job is to report what this evidence says about the question, even when it
is partial. Another stage decides whether your answer is well supported and
compares it against other readers' answers, so a partial or uncertain answer is
```

```
useful and a refusal is not. Refusing costs the system the answer entirely.

- If the evidence gives the answer, give it.
- If the evidence gives PART of the answer --- one of two values asked for, a
  related row, a number without its label, a trend without its magnitude ---
  report that part. Do not refuse because it is incomplete.
- If the evidence only lets you narrow it down, give the best-supported
  candidate and quote what you based it on.
- Use not_reported ONLY when the passage is about a clearly different subject
  than the question, or shares no relevant content at all. Uncertainty is not a
  reason to refuse; irrelevance is.

Rules:
- Use only the provided text and images. Never answer from general knowledge.
- ENTITY CHECK: identify the primary subject(s) of the question --- proper nouns
  (people, films, companies, datasets, models, papers, places, products). If the
  evidence describes a DIFFERENT entity (the question asks about
  Aaron Taylor-Johnson and the passage describes T.J. Miller's roles), that is
  the one case for not_reported: do not substitute a plausible-looking fact
  from the wrong entity. But evidence that mentions the right subject only
  briefly, or in a caption or a single table row, IS usable --- report what it
  says.
- For tables: when the question asks for a specific value, match the exact
  row/column intersection.
- For tables: when the question asks for improvements, gains, how much
  better, or increase over baseline, read ALL relevant rows (target model AND
  baseline rows) and compute the difference. Report delta values (e.g. +0.05),
  not absolute values.
- For which experiment/method/model: report the label, not the numbers.
- For which season/episode/edition/year/version performed best / is highest /
  is largest: report the LABEL (e.g. season two, 2015 edition), NOT the
  numeric value from its row.
- For questions that ask which of two named options is higher/better/superior
  (in A or B?, X or Y?, single-domain or multi-domain?, which setting
  achieves higher): extract the raw metric value for EACH named option, compare
  them directly, and name the winning option. Do NOT decide based on which
  option achieves new state-of-the-art relative to prior work --- compare the
  absolute values of the two options only.
- DIRECTION / POLARITY (critical): for questions about trends, changes, or
  relationships (increases/decreases, improves/worsens, higher/lower,
  positive/negative, rises/falls, correlates/inversely): (1) locate the exact
  values or explicit direction words in the evidence, (2) state the direction you
  read --- do NOT infer from expectation or general knowledge, (3) then give the
  answer. If the evidence shows X goes from 0.6 to 0.8, the direction is
  increases regardless of what is typical.
- BOOLEAN (critical): for yes/no questions, restate the exact relevant fact from
  the evidence before deciding. Never answer Yes/No from general knowledge --- only
  from the evidence in this context. If the evidence bears on the question but
  does not settle it, answer with the direction it points and quote the fact.

Respond with ONLY a JSON object, no other text:
{status:supported,answer:<exact answer>,evidence:<key phrase from text>}
or, only for a clearly different subject or no relevant content:
{status:not_reported,answer:,evidence:}
```

P5 Reconciliation (reasoner) Issued once per query, after the workers return. `$PATHS$` receives the per-chain drafts and the evidence behind them. It produces the answer that is scored.

P5 Reconciliation (reasoner) `config/prompts/reasoner_shape_trust.txt`

```
Answer the question from the evidence paths below.

Each path gives a **worker answer** and its **key evidence**. The worker read
the FULL passage for that path, including any images, tables and figures. The
key evidence below is a short text excerpt of that passage --- it is often
incomplete, and for slides, charts and tables it frequently does not contain
the number or label the worker read off the image.

QUESTION
$Q$

EVIDENCE PATHS
$PATHS$

How to weigh workers against excerpts:
```

```
1. A worker answer is wrong only if some key evidence CONTRADICTS it --- shows a
   different value, entity or direction for the same thing. An excerpt that is
   merely silent about the fact does not contradict the worker: the worker saw
   more than the excerpt shows. Never replace a worker's answer with not
   reported because you cannot see the fact in the excerpt yourself.
2. When several workers independently give the same answer, that agreement is
   strong evidence. Use it unless an excerpt directly contradicts it.
3. Where a worker refused but a path's key evidence contains the requested
   fact, extract it yourself.
4. Check the question's subject. Ignore paths whose key evidence is about a
   different entity, however plausible their answer looks.
5. Combine complementary partial findings across paths into one answer. If
   the question asks for several things, answer each; do not drop a part a
   worker supplied because another part is missing.
6. Only say the question cannot be answered if NO worker answered and NO
   excerpt contains the fact.

ANSWER FORM. Write the answer as a self-contained statement that a reader who
has not seen the question would understand: name the subject the question asks
about and state the fact. Never answer with a bare label, number, name, or
Yes/No on its own --- when a worker's answer is bare, complete it from the
evidence.
   - yes/no question: start with exactly Yes or No, then in the same
     sentence the fact that decides it.
   - names two options and asks which is higher/better: name the winner and
     give both values if the evidence has them, not Yes/No.
   - asks for a value: the subject, the value and its unit, exactly as written
     in the source.
   - asks how much something improved or differs: the signed difference, and
     the two absolute values it comes from.
   - asks which method/entity/item: the label, plus the property that makes it
     the answer (e.g. X, with the lowest error of 3.1%).
   - asks to list: every item the evidence supports, none dropped. If the
     question mentions a comparison or a reference, give both sides.
   - asks what a figure or table shows, what trend, relationship, correlation
     or observation it presents, or how one thing compares to another: state
     the direction of the relationship (what increases, decreases, or stays
     the same, and relative to what), name the quantities and the conditions,
     and then the reason if the evidence gives one. Two or three sentences.
     Never only describe what is plotted or which axes it has.
   - asks how, why, describe or explain: a short paragraph that covers each
     component the question names, grounded in the key evidence.
   Copy the source's exact wording for values, labels and names. Do not add
   facts the question did not ask about, and do not pad with generalities.

Wrap your reasoning in <thought>...</thought>, then your final answer in
<output>...</output>. The <output> tag must be the last thing you write, and
must contain no markdown, image references or nested tags.

<thought>
[2-5 sentences]
</thought>
<output>your answer</output>
```

P6 Sub-question aspect source Not part of the reported system. Used by the aspect-source ablation of Section 4.3, which replaces the spectral directions with generated sub-questions while holding the rest of the pipeline fixed.

P6 Sub-question aspect source `flowreader/prompts.py :: subquestion_prompt`

```
Break the question below into its distinct parts -- the separate things that
must each be found in the document for the question to be fully answered.

Rules:
- At most $N$ parts. Fewer is better; give one part if the question asks one
  thing.
- Each part must be answerable from the document on its own.
- Do not invent related questions. Only decompose what is actually asked.
- Keep each part short and concrete, in the question's own vocabulary.

Return JSON only: {parts: [..., ...]}

Question: $Q$
```

EVALUATION

Called once per query on the finished answer. No document content is shown to the judge.

P7 Judge Issued once per query over (question, gold answer, predicted answer), and used for every number reported in this paper. It is the evaluation prompt of Du et al. (2026), byte-identical to `config/prompts/eval.txt` in that work's repository, so our results and the baselines transcribed from it are scored by the same rubric as well as the same model and decoding settings (GPT-4o-mini, temperature 0, 300 max tokens).

P7 Judge `config/prompts/eval_g2reader.txt`

```
You are an expert evaluator tasked with assessing the accuracy of answers generated by a
  ↪ RAG
(Retrieval-Augmented Generation) system.

**Task**: Evaluate whether the generated answer correctly responds to the given question
  ↪ based on the expectedanswer.

**Question**: {question}

**Expected Answer**: {gold_answers}

**Generated Answer**: {assistant_answer}

**Evaluation Criteria**:
1. **Accuracy (0 or 1)**: Does the generated answer match the factual content of the
  ↪ expected answer?
- 1: The generated answer is factually correct and aligns with the expected answer
- 0: The generated answer is factually incorrect or contradicts the expected answer

**Instructions**:
- Focus on factual correctness, not writing style or format
- Consider partial matches: if the generated answer contains the correct information but
  ↪ includes additional context, it should still be considered accurate
- For numerical answers, check if the values match or are equivalent
- For list answers, check if all key elements are present
**Not answerable handling**:
  - If BOTH the expected answer AND generated answer indicate Not answerable or inability
  ↪ to answer, consider it accurate (accuracy=1)
  - If the expected answer provides specific content but the generated answer says Not
  ↪ answerable, it is INCORRECT (accuracy=0)
  - If the expected answer says Not answerable but the generated answer provides specific
  ↪ content, it is INCORRECT (accuracy=0)

**Output Format**:
Please respond with a JSON object containing only:
{{
accuracy: 0 or 1,
reasoning: Brief explanation of your evaluation
}}
- Output MUST be valid JSON (double quotes only).
- Do NOT use LaTeX, math delimiters, or any backslashes \ in reasoning.
- If you need to mention N_C, write it as N_C or NC (no \ characters).
- No code fences, no extra text.
```