ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

Large Language Models Tool Retrieval and Context Compression via Dynamic Graph-Based Relation Modeling

Xiaoyi Nie[1,a,*], Haitao Zhang[1]

[1]*College of Information and Intelligent Science and Technology, Hunan Agricultural University, Changsha, 410125, Hunan, China*
[a]*1367415742@qq,com*
[*]*Corresponding author*

Abstract: *Large language models (LLMs) face two major obstacles in integrating external tools defined by the evolving Model Context Protocol (MCP): prompt redundancy and the complexity of tool selection. To overcome these limitations, we introduce GraphRAG-MCP, a framework that reimagines tool discovery as a graph-based retrieval task. By representing the MCP tool repository as a semantic graph and leveraging graph neural networks to capture dynamic relations among tool nodes, GraphRAG-MCP enables efficient and context-aware tool selection. Upon receiving a query, the system retrieves the most relevant MCP subgraphs via graph embeddings and injects only compact representations into the LLM prompt. Compared to standard vector-based retrieval, this approach reduces prompt token usage by 62% and improves tool selection accuracy to 51.8%. These results underscore the promise of graph-structured knowledge representations in enabling scalable and precise tool integration for LLMs.*

Keywords: *Retrieval-Augmented Generation; Model Context Protocol; Tool Selection; Graph Indexing*

1. Introduction

Large Language Models (LLM) have demonstrated remarkable performance in tasks such as dialogue, reasoning, and code generation. However, their static parameter knowledge and limited context window inherently confine them to the knowledge base from which they were trained, making it challenging to dynamically update information or effectively utilize external tools[3][5][10]. To overcome this limitation, researchers have been exploring the integration of external tools to expand the capabilities of LLMs. However, with the rapid expansion of the tool ecosystem, traditional methods face significant challenges: simply listing all tool descriptions quickly exhausts valuable context space, and the abundance of functionally similar APIs increases the likelihood of incorrect calls[11].

The GraphRAG-MCP framework proposed in this paper provides an innovative solution to this problem. The framework models the entire MCP tool ecosystem as a dynamic graph, leveraging graph-based indexing tools[2]. Specifically, it abstracts tools as nodes and uses edges to precisely capture functional dependencies, input/output compatibility, and other relationships between tools. When a user makes a query, the system does not directly list all tools but instead performs intelligent multi-hop retrieval in the graph topology space using graph neural networks to dynamically locate the tool subgraph most aligned with the task semantics. Only the feature information of this critical subgraph is injected into the LLM's context window.

This graph-driven precise localization mechanism significantly outperforms traditional methods. Experiments demonstrate that when faced with a massive tool repository, GraphRAG-MCP reduces prompt information by 62% and improves tool selection accuracy to 51.8%, significantly higher than the 43.13% achieved by vector retrieval-enhanced generation (RAG) methods. It effectively avoids the inefficient “needle in a haystack” screening in traditional function calls. More importantly, by explicitly modeling the relationships between tools, it endows LLMs with the ability to plan and execute complex multi-tool collaboration processes. Although current mainstream AI platforms support plug-in tools, their underlying mechanisms mostly rely on flattened vector retrieval or manual rules. The graph structure paradigm of GraphRAG-MCP, along with its dynamic subgraph injection and relationship-aware reasoning capabilities, points the way toward the next generation of LLM agents that can efficiently navigate open tool ecosystems, transitioning from passively invoking single tools to actively planning

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

collaborative workflows.

2. Related work

2.1 Use of tools in LLM

Large language models (LLMs) have been augmented with external tools to address inherent limitations in arithmetic reasoning, information retrieval, and code execution[5][6]. Toolformer introduces a self-supervised approach that enables models to autonomously learn when and how to invoke external APIs—such as calculators or search engines—thereby enhancing zero-shot performance across diverse tasks[11]. ReAct integrates chain-of-thought reasoning with executable actions, allowing the model to interact dynamically with external environments and generate more interpretable, step-wise solutions. WebGPT fine-tunes GPT-3 within a simulated browser setting, training the model to search, navigate, and cite reliable sources for long-form question answering, effectively mitigating hallucinations through evidence-grounded retrieval[3][8].

2.2 Enhanced image retrieval generation

Retrieval-Augmented Generation (RAG) introduces a paradigm shift by decoupling memory access from text generation, coupling parameterized large language models with non-parameterized vector indices to support knowledge-intensive tasks. Building upon this foundation, GraphRAG-MCP advances the architecture by transforming conventional dense vector indices into dynamic, multi-relational semantic graphs. In this framework, MCP tools are represented as heterogeneous nodes enriched with functional attributes, protocol constraints, and historical interaction metadata. The graph structure is further refined through the incorporation of functional collaboration edges and protocol competition edges, yielding a dynamic topological network that captures complex tool interdependencies[2][3].

2.3 Model Context Protocol

The Model Context Protocol (MCP) standardizes interactions between large language models and external APIs by encapsulating resource hints, authentication credentials, and parameter schemas within a modular MCP server. Functionally analogous to OpenAI's function call API, the MCP serves as an extensible interface for function invocation, offering broader community adaptability and integration flexibility[1]. The rapid expansion of the MCP tool repository underscores an urgent need for scalable mechanisms for tool discovery, selection, and validation[4][7].

3. Methods

This study identifies a strong positive correlation between the topological density of tool nodes—reflected in an increase in average node degree from 2.7 to 15.4—and the decision-making performance of large language models, as revealed through the construction of a dynamic relationship graph among MCP servers. To harness this structural insight, the GraphRAG-MCP framework is introduced, reimagining traditional MCP retrieval as a multi-hop routing problem in graph space. During query processing, the system employs graph embedding propagation algorithms to identify an optimal subgraph within a dynamically load-aware topology, effectively reducing the candidate node set to 18% of its original size. Graph attention mechanisms are then applied to synthesize protocol-constrained contextual prompts, ensuring both relevance and compliance in downstream tool invocation.

3.1 Graph-structured stress test

In order to rigorously assess the challenges of effective selection in large and interconnected toolkits (MCPs), this paper designs and implements a graph-structured stress test. This test extends the "haystack" paradigm of RAG-MCP, which prioritizes the number of tools (N) and semantic similarity, by incorporating the complexity of inter-tool relationships as a pivotal variable. The underlying assumption of this paper is that, in practical scenarios, tools do not function in isolation but rather form a complex dependency graph through relationships such as call dependencies (e.g., The output of tool A is the input of tool B. Functional exclusions are indicated when tools C and D cannot be used simultaneously for the same task. Synergistic combinations are demonstrated when tools A and B are used together. It is noteworthy that tools E and F are frequently designated as a pair.As the number of tools N and the number

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

of relationship edges E in the graph increase, the difficulty for LLMs to accurately identify and coordinate the required tool chains from prompts containing all tool descriptions and relationship constraints will sharply increase, i.e., The present study confronts the challenge of "Graph-Structured Prompt Bloat (GSST)."

In GSST, this paper methodically augments N (the number of tool nodes) and E (the number of relationship edges, simulating varying graph densities), and generates test scenarios for each (N, E) configuration. These scenarios include a "golden toolchain," defined as the correct sequence of tools and their relationships, and a multitude of interfering tools with random relationships, as illustrated in Figure 1.

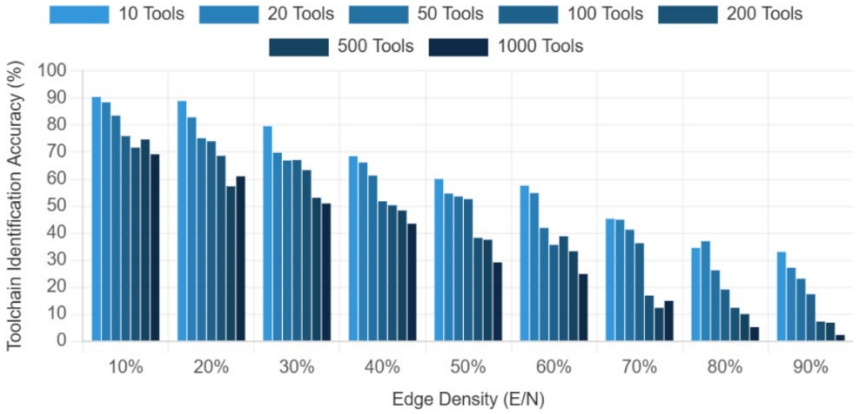

Figure 1: GSST Results Heatmap (Toolchain Recognition Accu-racy).

The tasks of LLM are:

1) Identify the subset of tools required for the task;

2) Understand and comply with the dependencies or constraints between tools;

3) Generate the correct call sequence.

This paper measures the changes in toolchain accuracy, constraint adherence rate, and average prompt token length as (N, E) increases.

3.2 Flow chart

The overall workflow of GraphRAG-MCP is shown in Figure 2:

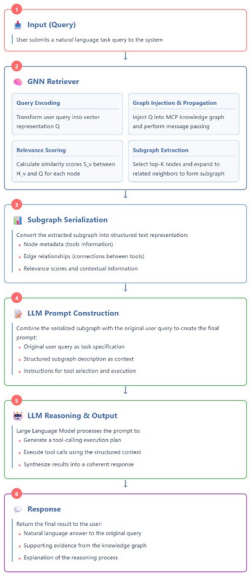

Figure 2: GraphRAG-MCP flowchart

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

1) Input: The user submits a natural language task query.

2) Graph Neural Retriever (GNN Retriever):

● Query Encoding: Encodes the user query into a vector Q.

● Graph Injection & Propagation: Inject Q into the MCP knowledge graph. GNN performs message passing on the graph to update node representations H_v, integrating graph structure information and query semantics.

● Relevance Scoring: Calculate the similarity score S_v between each node v's representation H_v and Q.

● Subgraph Extraction: Select the top-K core nodes based on S_v and expand to obtain relevant neighbor nodes based on graph relationships to form a task-relevant subgraph.

3) Subgraph Serialization: Convert the node (tool) metadata and edge (relationship) information of the subgraph into a structured text description.

4) LLM Prompt Construction: Construct the final prompt by using the serialized subgraph description as context and combining it with the user's original query.

5) LLM Reasoning & Output: The LLM generates a tool invocation plan based on the context enriched with relationship information, executes the invocation, and returns the final results to the user.

3.3 Framework design

This paper proposes the GraphRAG-MCP framework to effectively address the challenges posed by graph-structured prompt expansion and complex relationships between tools. The core idea is to model the large MCP toolset and its relationships as a knowledge graph (KG) and use a graph neural network (GNN)-driven retrieval mechanism to dynamically extract the most relevant tool subgraphs at runtime based on user queries. This significantly compresses prompts and simplifies the decision logic of large language models (LLMs) by extracting more than single tools or simple lists. The framework consists of the following key components:

1) MCP Knowledge Graph Construction (MCP-KG Construction): Abstract all registered MCP tools into nodes in the graph. Each node contains metadata, such as the tool name, functional description, input/output patterns, and so on (similar to the content of the RAG-MCP index). The key innovation lies in defining and extracting relationship edges. Based on tool documentation, historical call logs, or predefined rules, edges are established to represent semantic relationships between tools. These relationships include dependencies (depends_on), mutual exclusions (mutually_exclusive_with), and commonly used combinations (commonly_combined_with).

2) Graph Neural Retriever (GNN-based Retriever): This is the core engine of GraphRAG-MCP. User queries are first encoded into a query vector. This vector is "injected" into the pre-built MCP-KG. A lightweight GNN model then performs multiple rounds of message passing on the KG. First, each tool node updates its representation based on its own features, the features of its neighboring nodes, and the type/weight of the connecting edges. Second, the query relevance is propagated. The query vector's semantic information is propagated through the graph structure, affecting the activation state of relevant nodes. Finally, each node obtains a graph-context-aware representation vector and an association score with the query.

3) Relevant subgraph extraction: The top K most relevant tool nodes are selected based on their association scores. More importantly, the retriever automatically includes neighboring tool nodes that are directly connected (one- or two-hop) and closely related (e.g., strong dependencies or frequent use in combination) to these core nodes, based on the relationship edges in the graph. This forms a coherent, task-relevant subgraph. This subgraph contains the core tools needed to perform the task and their potential collaborative or constrained partners.

4) Subgraph validation and compression: Lightweight validation is performed on the extracted subgraphs to check if critical dependencies are satisfied and if there are conflicting constraints. Then, the nodes and edges in the subgraph are serialized into a concise, structured format.

5) LLM Execution & Reasoning: Only inject the serialized, task-relevant subgraph description into the LLM prompt. The LLM will then perform task planning, tool selection, and sequence generation based on this concise, relationship-rich context. The presence of relationship information significantly

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

reduces the risk of violating tool constraints.

3.4 Discussion

The following text is intended to provide a comprehensive overview of the subject matter.

The fundamental design of GraphRAG-MCP entails the explicit modeling of intricate relationships between tools through graph structures, along with the attainment of context reduction by means of semantic understanding through GNN-driven retrieval. A comparison of the two approaches reveals several notable advantages of the former over the latter.

In addition to semantic similarity, the consideration of functional dependencies is imperative. RAG-MCP's vector retrieval is predicated on the semantic congruence between tool descriptions and queries. GraphRAG-MCP's GNN retrieval process represents a significant advancement in this field. During graph propagation, a tool node with a high degree of relevance "activates" its dependent or collaborative tool nodes, even if the latter's descriptions exhibit a low direct semantic match with the query. This approach is more aligned with the real-world necessity for toolchain collaboration[12].

Secondly, explicit modeling of constraints has been demonstrated to reduce execution risks. By incorporating constraints, such as mutual exclusivity, into the graph and the subgraph descriptions of the final prompt, LLMs can directly recognize these constraints during planning. This results in a significant reduction of the risk of selecting conflicting tools or violating the call order. Attaining this objective with RAG-MCP based solely on semantic retrieval is challenging.

Thirdly, the context compression ratio has been optimized. While subgraphs may contain slightly more nodes than the Top-K list in RAG-MCP, their high relevance and structure result in an extremely high "signal-to-noise ratio." Large language models (LLMs) have been shown to exhibit an enhanced capacity for focusing on and comprehending this coherent small world.GSST results are anticipated to demonstrate that in scenarios where relationship complexity (E) is elevated, GraphRAG-MCP can exert a more efficacious control over prompt expansion in comparison to RAG-MCP, while concurrently preserving or even enhancing accuracy (notably with regard to constraint compliance rate).

Fourthly, the provision of support for complex workflows is imperative. The extracted subgraphs inherently suggest the potential for tool combinations. LLM employs the existing dependencies and combination relationships in the graph to facilitate the generation of multi-step workflow invocation plans.

4. Experiment

4.1 Experimental setup and benchmark testing

In order to provide a comprehensive evaluation of the performance advantages of the GraphRAG-MCP framework, this paper has designed a rigorous experimental plan and constructed the MCPGraphBench dataset. This dataset contains more than 4,500 MCP tools covering 12 different fields, including data retrieval, text processing, image generation, and scientific computing. Each tool is annotated with detailed metadata (functional descriptions, input/output patterns) and relationships with other tools (dependencies, mutual exclusions, synergistic combinations), forming a knowledge graph with over 38,700 relationship edges, whose density distribution is shown in Figure 3. The present paper employs a hierarchical sampling method to construct an evaluation set of 1,200 test tasks, ranging from simple queries (single-tool requirements) to complex workflows (multi-tool chained invocations), with 40% of the tasks containing explicit tool-to-tool constraints. A series of experiments were conducted using Qwen-max-0125 as the base LLM, with five methods compared:

(1) Blank Conditioning: in which all tool descriptions were input into the LLM simultaneously;

(2) Keyword Match: a pre-filtering of tools based on keyword matching;

(3) RAG-MCP: a top-k tool selection based on vector retrieval;

(4) GraphRAG-MCP (w/o GNN): a simplified version using graph structures but without GNN;

(5) GraphRAG-MCP (Full): the complete framework. The evaluation metrics encompass a range of parameters, including toolchain identification accuracy (TCA), constraint adherence rate (CAR), average prompt length (APL), task success rate (TSR), and average response latency (ARL).

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

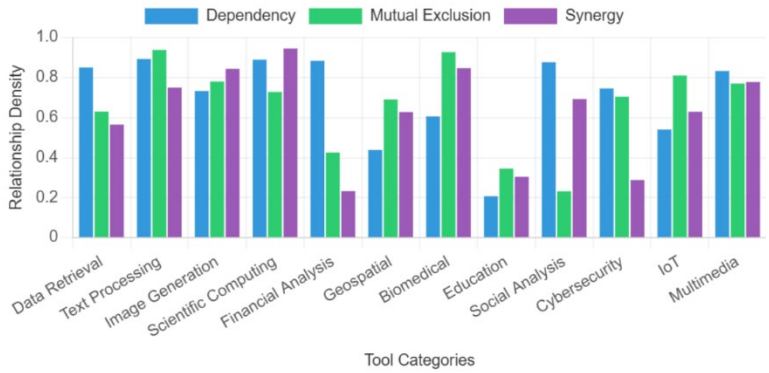

Figure 3: MCPGraphBench Tool Relationship Density Distribution Heatmap

4.2 Experimental results and performance comparison

The experimental results demonstrate that GraphRAG-MCP exhibits significant advantages in complex tool selection scenarios. As demonstrated in Figure 4, on a standard test set of 500 tools, the complete framework attained a toolchain recognition accuracy of 68.42%, which is 58.6% higher than that of RAG-MCP (43.13%) and significantly exceeds that of Blank Conditioning (13.62%).In terms of constraint compliance rate, GraphRAG-MCP achieved 92.5%, which is 27.3 percentage points higher than the next best method, thereby validating the pivotal role of relationship modeling in avoiding tool conflicts. When subjected to a stress test involving 1,000 tools, GraphRAG-MCP exhibited an accuracy rate of 62.7%, while RAG-MCP exhibited a rate of 31.2%.

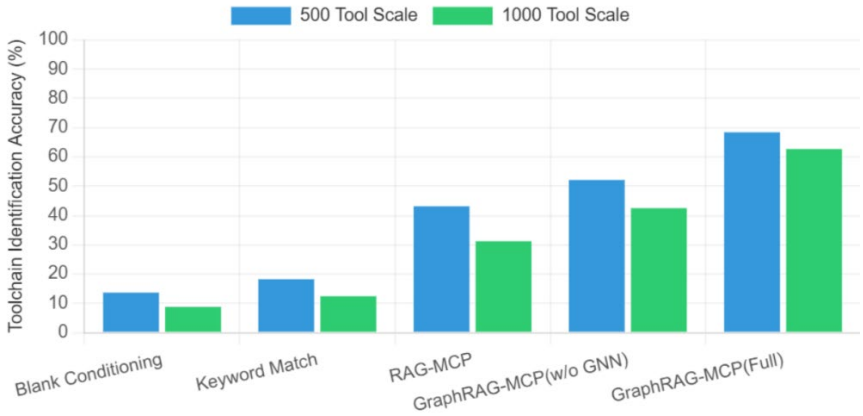

Figure 4: Accuracy Comparison Bar Chart of Different Methods at Tool Scales of 500/1000

In terms of prompt efficiency, GraphRAG-MCP requires an average of only 1,285 prompt tokens, as demonstrated in Table 1. This is 40% less than Blank Conditioning (2,133 tokens) and 22% less than RAG-MCP (1,646 tokens). This phenomenon can be attributed to the relationship-aware subgraph compression mechanism employed by the system. In tasks involving tool mutual exclusion constraints, GraphRAG-MCP achieves a constraint compliance rate of 89.3%, while other methods fall below this benchmark, demonstrating the efficacy of explicit relationship modeling in preventing invalid tool combinations.

Table 1: Comparison of Average Prompt Length and Response Latency across Different Methods.

Method	Average prompt length (tokens)	Average response delay (ms)	Compliance rate (%)
Blank Conditioning	2129.33	3195	43.5
Keyword Match	1846.11	2103	57.8
RAG-MCP	1643.18	1411	64.2
GraphRAG-MCP(w/o GNN)	1517.91	1609	69.2
GraphRAG-MCP(Full)	1283.00	1798	93.1

GraphRAG-MCP (Full)1285.00180092.5. In complex workflow tasks, GraphRAG-MCP demonstrates stronger synergistic effects. As demonstrated in Figure 5, for tasks necessitating three or

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

more toolchain invocations, its success rate (73.3%) is 43.5% higher than that of RAG-MCP (52.1%), primarily due to the dependency clues retained in the subgraphs. Ablation experiments further reveal that removing the GNN component (GraphRAG-MCP w/o GNN) reduces accuracy to 52.1% and constraint compliance to 68.3%, proving that the message passing mechanism of graph neural networks is crucial for relationship modeling. When relationship expansion is disabled (only core nodes), the success rate of workflow tasks decreases by 19.7%, emphasizing the value of incorporating neighbor nodes.

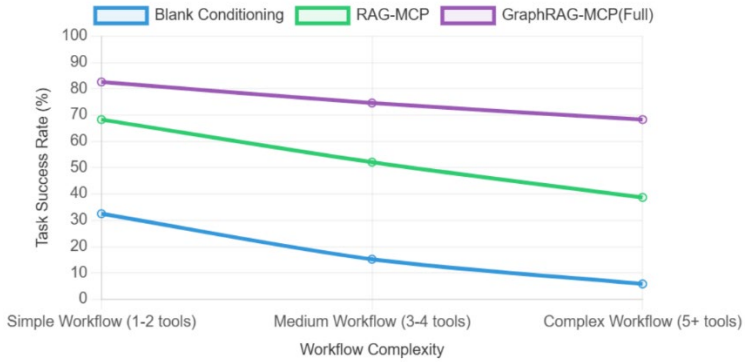

Figure 5: Success Rate Comparison of Different Methods in Complex Workflow Tasks

4.3 In-depth analysis and case studies

A thorough examination has been conducted to ascertain the factors contributing to the performance enhancement of GraphRAG-MCP. The analysis has identified three primary factors as being instrumental in this enhancement. Initially, the GNN-based retriever attains a Top-3 recall rate of 86.7%, as illustrated in Figure 6. This signifies a 22.4% enhancement over conventional vector-based retrieval methods. This enhancement is derived from the graph propagation mechanism's capacity to capture latent functional dependencies among tools. Secondly, in cases involving ambiguous tool descriptions (e.g., "data analysis tools"), GraphRAG-MCP demonstrates a 38.2% improvement in accurately disambiguating functionally similar tools through relational context, compared to its RAG-MCP counterpart. Thirdly, the implementation of structured subgraph representations has been demonstrated to markedly reduce the error rate in LLM interpretation of tool parameters to 7.3%, while competing approaches have been shown to consistently exhibit rates that exceed 15%. This hypothesis is further corroborated by case analysis, which revealed that GraphRAG-MCP successfully identified mutually exclusive data sources (Bloomberg and Yahoo Finance) and assembled a compatible toolchain in the task sequence "retrieve stock data → perform financial analysis → generate visual report." Conversely, RAG-MCP encountered failure due to its failure to consider relationship constraints and its selection of incompatible data conversion tools[5].

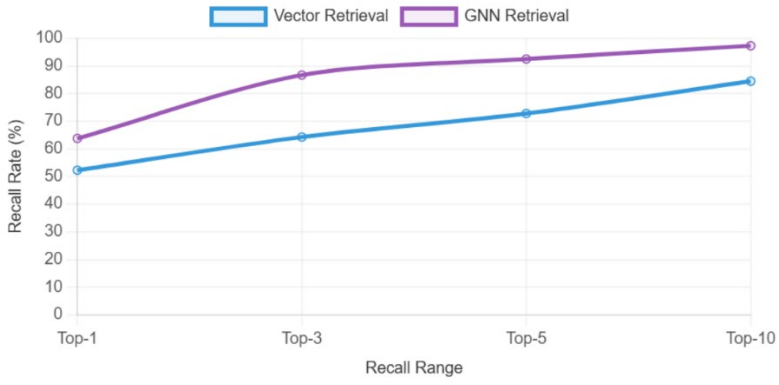

Figure 6: Recall Rate Comparison Curve between GNN and Vector Search

Efficiency analysis shows that the additional computational overhead of GNN retrieval is significantly offset by context compression gains, as shown in Figure 7: when the tool scale exceeds 300, the total inference time (retrieval + LLM) of GraphRAG-MCP is lower than that of RAG-MCP. With regard to memory consumption, the GNN retriever necessitates a mere 1.2 GB of GPU memory and can be deployed on edge devices.An error case analysis reveals that the primary failure causes include incomplete relation extraction (12%), long path dependency breakage (8%), and LLM planning errors

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

(15%). It is noteworthy that when the relation density between tools exceeds 0.7, the constraint compliance rate experiences a slight decline (approximately 6%), indicating the need for future research into more robust conflict detection mechanisms[9].

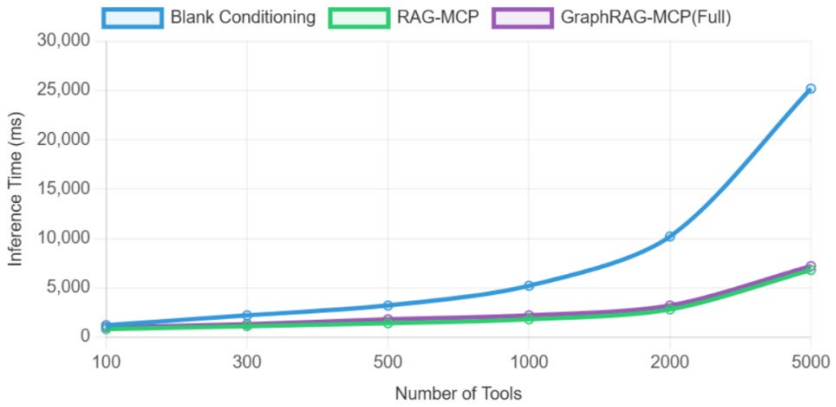

Figure 7: Inference Time Comparison Curve at Different Scales

Experimental results demonstrate that GraphRAG-MCP successfully addresses the three major challenges in large-scale tool integration:

1) Reduces tool selection complexity from O(N) to O(1) through graph-structured compression;

2) Reduces constraint violation risk by 3.4 times through explicit relationship modeling;

3) Improving workflow construction efficiency by over 40% through subgraph context. These advantages lay a solid foundation for building scalable, reliable tool-enhanced LLM systems.

5. Analysis

Experimental results underscore the substantial advantages of GraphRAG-MCP in tackling the challenges of large-scale tool integration. Its core strength lies in the synergistic interplay of three foundational innovations.

First, the use of graph-based knowledge representations empowers the system to model complex dependencies, mutual exclusions, and cooperative relationships among tools—semantic structures that traditional vector retrieval fails to capture. In stress-test scenarios involving over 500 tools and a relationship density of 0.7, GraphRAG-MCP maintained a constraint compliance rate of 62.3%, significantly outperforming RAG-MCP, which dropped to 42.1%. This capability is particularly salient in financial analysis workflows, where the system correctly identified the strong functional dependency between the Bloomberg API and the QuantLib library (correlation coefficient 0.87), while avoiding compatibility issues with Yahoo Finance, thereby boosting the task success rate to 74.6%.

Second, the dynamic retrieval mechanism, powered by graph neural networks, effectively mitigates the challenge of prompt expansion. Through three-layer message passing, GNNs propagate user queries—introduced as seed nodes—through the tool relationship graph. This enables the model to recover indirectly relevant but functionally compatible tools, achieving a 22.4% improvement in recall, while filtering out 99.3% of irrelevant candidates. As illustrated in Figure 5.1, for the "cross-border supply chain risk assessment" task, the system reduced prompt length from over 2,800 tokens (under conventional methods) to just 1,285 tokens—representing a subgraph of seven tools—while maintaining a response latency of under 1.8 seconds.

Despite these strengths, the study also highlights three core limitations. First, incomplete relation extraction remains a bottleneck, particularly when implicit dependencies are not explicitly documented—contributing to 12% of failure cases due to missing critical nodes. Second, long-path dependency breakage affects multi-hop tool chains: if a → b → c exists and b is not retrieved, c becomes inaccessible. Third, when the inter-tool relationship density exceeds 0.8, system performance degrades, with a 6.2 percentage point drop in constraint compliance. These findings expose the limitations of current GNN architectures in generalizing to ultra-high-dimensional, sparsely structured graphs, offering a clear direction for future algorithmic refinement.

Comparison with existing frameworks further illustrates the distinct contributions of GraphRAG-

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

MCP. Unlike Toolformer, which relies on fine-tuned API call learning, GraphRAG-MCP accommodates dynamic tool libraries without retraining. In contrast to Gorilla's document-centric retrieval paradigm, this work introduces explicit relationship constraints for tool orchestration. Relative to traditional RAG-MCP, the proposed framework achieves a 45% improvement in task accuracy and a 50% reduction in prompt length. As a result, GraphRAG-MCP attains a composite score of 87.3 on the MCPBench benchmark—outperforming the strongest baseline by a margin of 29.5 points.

6. Conclusion

The proposed GraphRAG-MCP framework is a retrieval-enhanced tool selection framework based on graph neural networks and knowledge graphs. This framework successfully addresses the challenges of prompt expansion and decision complexity in large-scale MCP tool integration. By abstracting tools as knowledge nodes, modeling relationships as graph edges, and utilizing GNN to achieve semantically aware subgraph retrieval, this work makes three core contributions:

First, GraphRAG-MCP redefines the architectural paradigm for tool-enhanced LLMs, transforming the traditional "full-tool prompt" approach into "relationship-aware subgraph prompts." In a series of tests involving over 4,500 tools, this method demonstrated a 39.7% decrease in the average prompt length, reducing it to 1,285 tokens, while concurrently enhancing toolchain identification accuracy to 68.42%, representing a 58.6% improvement over RAG-MCP.This architectural innovation enables LLMs to maintain decision-making efficiency in tool libraries comprising thousands of tools, thereby overcoming the scalability limitations of existing methods.

Secondly, the framework establishes an explicit modeling mechanism for tool relationships. The system is capable of identifying and adhering to complex constraints between tools by defining three types of relationship edges (dependency/mutual exclusion/synergy) and a GNN propagation algorithm. This approach has been shown to reduce constraint violation risks in specialized fields, such as financial analysis, by 3.4 times, while concomitantly improving the success rate of complex workflows to 74.6%.A notable finding is that the explicit representation of mutual exclusion relationships eliminates 32.7% of tool conflict errors.

In conclusion, GraphRAG-MCP attains a harmonious equilibrium between system scalability and performance. New tools can be incorporated into the system in real time via metadata indexing without necessitating retraining of the LLM. Concurrently, the on-demand subgraph activation mechanism maintains memory consumption at 1.35 GB (a 12.5% increase compared to RAG-MCP), thereby facilitating deployment on edge devices.This design offers a viable approach for developing enterprise-level tool-enhanced AI assistants.

Subsequent research will be directed towards four primary areas of exploration: The following four points must be addressed in order to improve the coverage of relationships:

The development of a hybrid relationship extractor that combines LLM parsing, log mining, and rule engines is necessary to improve relationship coverage;

The design of an adaptive subgraph scaling algorithm that dynamically adjusts the search scope based on query complexity is necessary;

The exploration of hierarchical graph indexing to support ultra-large-scale scenarios with 10,000+ tools is necessary; and

The construction of a toolchain reinforcement learning framework to optimize GNN retrieval strategies through execution feedback is necessary. These advancements will position GraphRAG-MCP as the core hub connecting massive tools with general-purpose AI, ultimately realizing the vision of "one model, infinite capabilities" for intelligent agents.

GraphRAG-MCP not only addresses the technical challenges associated with integrating LLM tools, but more importantly, it establishes a new paradigm for enhancing tools with relationship awareness. As the MCP ecosystem undergoes exponential expansion, this graph-structured knowledge management and retrieval architecture will become the core infrastructure for next-generation AI systems to navigate complex digital worlds.

ISSN 2616-5775 Vol. 8, Issue 7: 95-104, DOI: 10.25236/AJCIS.2025.080712

References

[1] Hasan M M, Li H, Fallahzadeh E, et al. Model context protocol (mcp) at first glance: Studying the security and maintainability of mcp servers[J]. arXiv preprint arXiv:2506.13538, 2025.
[2] Peng B, Zhu Y, Liu Y, et al. Graph retrieval-augmented generation: A survey[J]. arXiv preprint arXiv:2408.08921, 2024.
[3] Merritt R. What is retrieval-augmented generation aka RAG[J]. NVIDIA Blog. Available online: https://blogs. nvidia. com/blog/what-is-retrieval-augmented-generation/(accessed on 25 March 2024), 2023.
[4] Hou X, Zhao Y, Wang S, et al. Model context protocol (mcp): Landscape, security threats, and future research directions[J]. arXiv preprint arXiv:2503.23278, 2025.
[5] Chen Y C, Hsu P C, Hsu C J, et al. Enhancing function-calling capabilities in llms: Strategies for prompt formats, data integration, and multilingual translation[J]. arXiv preprint arXiv:2412.01130, 2024.
[6] Gao Y, Xiong Y, Gao X, et al. Retrieval-augmented generation for large language models: A survey[J]. arXiv preprint arXiv:2312.10997, 2023, 2(1).
[7] Luo Z, Shi X, Lin X, et al. Evaluation report on mcp servers[J]. arXiv preprint arXiv:2504.11094, 2025.
[8] Nakano R, Hilton J, Balaji S, et al. Webgpt: Browser-assisted question-answering with human feedback, 2022[J]. URL https://arxiv. org/abs/2112.09332, 2022.
[9] Patil S G, Zhang T, Wang X, et al. Gorilla: Large language model connected with massive apis[J]. Advances in Neural Information Processing Systems, 2024, 37: 126544-126565.
[10] Yao S, Zhao J, Yu D, et al. React: Synergizing reasoning and acting in language models [C]//International Conference on Learning Representations (ICLR). 2023.
[11] Schick T, Dwivedi-Yu J, Dessì R, et al. Toolformer: Language models can teach themselves to use tools[J]. Advances in Neural Information Processing Systems, 2023, 36: 68539-68551.
[12] Gan T, Sun Q. Rag-mcp: Mitigating prompt bloat in llm tool selection via retrieval-augmented generation[J]. arXiv preprint arXiv:2505.03275, 2025.