LLM-GMP: Large Language Model-Based Message Passing for Zero-Shot Learning on Graphs

Justin Carpenter[*]
Computer Science Department
Boise State University
Boise, Idaho, USA
JustinCarpenter836@u.boisestate.edu

Md Athikul Islam[*]
Computer Science Department
Boise State University
Boise, Idaho, USA
mdathikulislam@u.boisestate.edu

Edoardo Serra
Computer Science Department
Boise State University
Boise, Idaho, USA
edoardoserra@boisestate.edu

***Abstract*—Graph-structured data is ubiquitous to many scientific and industrial applications, with tasks such as node classification, edge prediction, and graph classification widely studied using Graph Neural Networks (GNNs). While effective, GNNs rely on supervised training and struggle with zero-shot learning, where models must generalize to unseen tasks without labeled data.**

Recent work has explored Large Language Models (LLMs) for zero-shot graph reasoning by converting graphs into text, but these approaches are hindered by context window limits, scalability issues, and hallucinations.

We propose Large Language Model Graph Message Passing (LLM-GMP), a framework that reformulates message passing as task-aware textual communication between nodes. This design enables scalable, interpretable, and distributed reasoning with LLMs. We evaluate LLM-GMP across diverse benchmark datasets and demonstrate its effectiveness against a range of state-of-the-art baselines.

LLM-GMP achieves competitive or superior zero-shot accuracy, in some cases approaching and outperforming supervised GNNs, while requiring no training. Runtime analysis further shows that scalability depends on graph size and density, with structured message passing improving efficiency.

By aligning graph exploration with LLM reasoning, LLM-GMP establishes a new paradigm for interpretable and scalable zero-shot graph learning at the intersection of GNNs, LLMs, and agentic AI.

***Index Terms*—Large Language Models (LLMs), Graph Neural Networks (GNNs), Zero-Shot Learning, Message Passing**

I. Introduction

Graphs offer a natural and powerful way to represent data with relational and structural dependencies, making them essential in domains such as social network analysis, biological systems, knowledge graphs, and recommendation engines [4], [14], [26]. The advent of GNNs has significantly advanced the state of the art in learning on graph-structured data [19], [28]. By employing message-passing mechanisms that iteratively aggregate information from neighboring nodes, GNNs enable expressive and scalable representations for a variety of tasks, including node classification, link prediction, and graph classification [28], [31].

However, despite these advances, zero-shot learning (ZSL) on graphs remains a persistent challenge [15]. In the zero-shot setting, models are required to generalize to previously unseen tasks or label spaces without additional labeled training data [2]. Traditional GNNs are ill-equipped for this scenario: they typically rely on supervised training tailored to specific tasks and lack the adaptability to handle new objectives without retraining or architectural modifications.

In parallel, LLMs have recently demonstrated impressive zero-shot generalization across a wide range of NLP and reasoning tasks, thanks to pretraining on massive and diverse text corpora [11], [23]. Inspired by this capability, researchers have begun exploring the use of LLMs for graph problems, primarily by translating graph structures into textual formats that LLMs can process [3], [23], [24]. While this approach has shown promise—especially in zero-shot contexts—it is fundamentally limited. Flattening entire graphs into text often exceeds the LLM's context window, particularly for large or densely connected graphs, and can introduce semantic ambiguity or information loss. Moreover, these methods are prone to hallucinations, where the model generates outputs that deviate from the underlying graph structure, undermining reliability in critical applications.

To address these limitations, we propose Large Language Model Graph Message Passing (LLM-GMP), a novel framework that reimagines graph processing through the lens of zero-shot language-based reasoning. Rather than representing the graph as a static text block, LLM-GMP simulates message passing in the language domain: nodes exchange task-aware, interpretable text

[*]These authors contributed equally to this work.

messages generated by an LLM. These messages evolve over multiple rounds of interaction, allowing information to propagate across the graph in a structured, hierarchical manner. Crucially, the LLM is aware of the downstream task throughout this process, enabling it to synthesize and interpret contextual information effectively.

More concretely, LLM-GMP defines a message passing algorithm in which each node iteratively aggregates textual information received from its neighbors, with the LLM generating messages tailored to the specific zero-shot learning task. Over successive iterations, the LLM refines each node's understanding of its local and global context, culminating in task-specific reasoning (e.g., classification or prediction) based on the final aggregated messages. This paradigm harnesses the complementary strengths of message passing and LLM-based inference, achieving both interpretability and flexibility without retraining.

To the best of our knowledge, this is the first framework to define such a paradigm, which can be viewed as a form of agentic AI, where each node acts as a lightweight reasoning agent within a collaborative, task-driven system. Preliminary experiments show that LLM-GMP offers strong performance across a range of zero-shot graph learning tasks, highlighting its potential as a robust alternative to traditional graph representation learning approaches.

Our main contributions are:

- We introduce LLM-GMP, the first framework that leverages LLMs for zero-shot graph learning via iterative message passing, combining interpretability with flexibility.
- We design a scalable message passing scheme that mitigates LLM limitations, such as context window size and hallucinations by structuring inference as localized node-level interactions.
- We conduct extensive experiments on 9 benchmark networks, showing that LLM-GMP either outperforms baselines or achieves comparable results to supervised GNNs without requiring labels.
- We provide a runtime analysis demonstrating how graph size and density influence computational cost, and discuss strategies for future optimization, such as batching and distributed inference.

II. Related Work

This section reviews three key research directions relevant to our work: *graph representation learning*, *graph neural networks*, and the emerging line of *LLMs operating directly on graphs*.

A. Graph Representation Learning (GRL)

Graph representation learning (GRL) aims to embed nodes into low-dimensional vectors that preserve both structural and attribute information, enabling tasks such as classification, clustering, and community detection. Early dimensionality reduction techniques, including Local Linear Embedding and Laplacian Eigenmaps, established the foundation but suffered from scalability limitations.

A major breakthrough came with random-walk–based methods. DeepWalk [17] generates truncated random walks to treat node sequences analogously to sentences, using language models to learn embeddings that capture neighborhood similarities. Node2vec [6] extends this idea with biased random walks, balancing exploration of local and global structural properties. Attribute-aware variants further incorporate node features during walks or matrix factorization, enriching embeddings with semantic information. Matrix factorization methods, in turn, decompose adjacency or feature matrices directly, with inductive adaptations supporting textual attributes.

GRL also expanded into semi-supervised and unsupervised paradigms. Semi-supervised GRL integrates partial labels via regularization or context prediction, with GCN-based approaches excelling by propagating label signals across layers while fusing topology and attributes. Unsupervised variants such as graph autoencoders reconstruct structural information for clustering. More recent hybrids combine GCNs with Markov random fields or community-aware strategies to improve performance [25], though often at increased computational cost.

B. Graph Neural Networks

GNNs learn node representations by aggregating information from local neighborhoods. Graph Convolutional Networks (GCN) [10] introduced an efficient layer-wise propagation rule that approximates spectral graph convolutions. In practice, a GCN layer updates a node's representation by taking a weighted average of its own features and those of its immediate neighbors, effectively smoothing features across the graph. Graph Attention Networks (GAT) [22] improved upon this by incorporating self-attention. Instead of using fixed, static weights for neighbors, like GCN. GAT learns to assign different importance weights to different neighbors, allowing the model to focus on the most relevant information in a node's vicinity. Graph Isomorphism Networks (GIN) [29] uses a simple yet powerful update rule, summing neighbor features and processing the result through a Multi-Layer Perceptron. This method was shown to be as powerful as the Weisfeiler-Lehman Isomorphic test. [27] Despite their success, all these models are supervised and depend on labeled data and therefore are not inherently equipped for zero-shot learning tasks.

Algorithm 1 LLM-Guided Message Passing with Iteration-wise Aggregation and Parallel Batching

```
Input: Graph $G = (V, E)$, LLM model LLM, number of message steps / iterations $T$, node batch size $b$
 1: for all node $v_i \in V$ do
 2:     $h_i^{(0)} \leftarrow \text{OriginalRepresentation}(v_i)$
 3: end for
 4: for iteration $t = 1$ to $T$ do
 5:     Partition nodes $V$ into non-overlapping batches $\{B_1, B_2, \ldots, B_K\}$ of size $b$
 6:     for all batch $B_k$ do
 7:         Step 1: Construct prompts for all nodes in the batch
 8:         for all node $v_i \in B_k$ do
 9:             $\text{prompt}_i \leftarrow \text{ConstructPrompt}\big(v_i, h_i^{(t-1)}, \{h_j^{(t-1)} \mid v_j \in \mathcal{N}(i)\}\big)$
10:         end for
11:         Step 2: Evaluate prompts jointly with LLM
12:         $\{h_i^{(t)} \mid v_i \in B_k\} \leftarrow \text{LLM}(\{\text{prompt}_i \mid v_i \in B_k\})$
13:     end for
14: end for
15: for all node $v_i \in V$ do
16:     $\hat{y}_i \leftarrow \text{LLM or Classifier}(h_i^{(T)})$
17: end for
18: return Predicted labels $\{\hat{y}_i\}_{i \in V}$
```

C. LLMs Operating Directly on Graphs

The integration of large language models (LLMs) into graph processing has gained considerable traction, showcasing their capacity to reason over complex graph-structured data. Li et al. [12] investigated the structural analysis abilities of LLMs, introducing specialized benchmarks and datasets to support rigorous evaluation. Fan et al. [5] demonstrated that LLMs can effectively augment traditional GNNs, enhancing their representational power and predictive performance. Tang et al. [21] proposed GraphGPT, an instruction-tuned framework that fuses LLMs with graph knowledge, enabling robust generalization across diverse graph datasets.

A parallel line of work focuses on the integration of LLMs with knowledge graphs (KGs). Ibrahim et al. [9] categorized the landscape into three main paradigms: KG-augmented LLMs, LLM-augmented KGs, and hybrid frameworks that combine both methodologies.

Community-driven initiatives, such as the "Awesome-Graph-LLM" repository [8], have emerged to catalog ongoing research and tools at the intersection of graphs and LLMs. Benchmarking platforms like GraphEval36K [20]—which includes 40 graph-related coding tasks and 36,900 test cases—provide comprehensive assessments of LLMs' reasoning capabilities in graph domains. Similarly, GPT4Graph [7] evaluates how well LLMs understand and manipulate graph-structured data, shedding light on their strengths and limitations.

Notably, the most effective zero-shot approaches for graph tasks are those that translate graph structures into textual representations, making them directly accessible to LLMs. However, these methods are constrained by the limited context window of current models and are susceptible to hallucinations, which may affect reliability in critical applications.

III. Methodology

The methodology comprises LLM-Guided Message Passing, incorporating Batch-wise Aggregation and tailored LLM Prompt Design to enable effective graph-level reasoning.

A. LLM-Guided Message Passing with Iteration-wise Batching

We formulate our task as a *node classification* problem, where the goal is to predict the class label of each node in a graph. Inspired by the *message passing* paradigm in GNNs, we reimagine node communication using LLMs, which offer semantically rich and adaptive reasoning capabilities. Instead of applying static aggregation functions across all neighbors in a single pass, we introduce an iterative message-passing approach with batching for efficient computation.

1) Iteration-wise Batching Strategy: Conventional GNNs aggregate information from all neighboring nodes simultaneously, typically using operations such as summation, averaging, or attention mechanisms. While effective for numerical features, such approaches often struggle to capture nuanced semantic interactions when applied to textual graphs. To address this limitation, we leverage LLM-guided message passing, in which each node's representation is refined iteratively by aggregating information from *all neighbors*, while batching is used solely for parallel processing of LLM queries.

Let V be the set of nodes in the graph, and $v_i \in V$ a node with neighborhood $\mathcal{N}(i)$. At each message-passing step t (for a total of T steps), the nodes V are partitioned into non-overlapping batches $B_1, B_2, \ldots, B_K$ of size b to reduce computational overhead. Within each batch B_k, we first construct prompts for all nodes in parallel:

$$\text{prompt}_i = \text{ConstructPrompt}\Big(v_i, h_i^{(t-1)}, \{h_j^{(t-1)} \mid v_j \in \mathcal{N}(i)\}\Big) \,;\forall v_i \in B_k$$

These prompts are then jointly evaluated by the LLM to produce updated representations for all nodes in the batch:

$$\{h_i^{(t)} \mid v_i \in B_k\} = \text{LLM}(\{\text{prompt}_i \mid v_i \in B_k\})$$

This process continues across all batches in each iteration and for all T message-passing steps. Node batching thus enables efficient parallel LLM calls, while ensuring that each node still incorporates the complete information from its neighborhood.

The advantages of this approach include: (i) efficient parallel computation without sacrificing neighbor information, (ii) interpretable and traceable updates at each iteration, and (iii) enriched contextual reasoning, as the LLM balances the node's original content with information aggregated from its neighborhood.

The complete procedure is formalized in Algorithm 1, which outlines initialization, batch construction, prompt generation, and iterative refinement of node representations. This design ensures scalability, reproducibility, and seamless integration with transformer-based language models for graph-centric tasks.

B. LLM Prompt Design

The effectiveness of our message passing framework relies critically on the design of natural language prompts that guide the LLM. We construct two types of prompts: one for iterative message passing updates and one for final node classification. These prompts are informative, modular, and context-aware, allowing the LLM to process localized information while remaining grounded in the overall task objective.

Unlike classical GNNs, where feature aggregation is handled numerically, our approach uses text instructions to stimulate semantic reasoning in the LLM. Each prompt explicitly encodes the task type, iteration context, and relevant input segments, such as the original node text, the current semantic state, and the texts of neighboring nodes. This design ensures the LLM preserves semantic grounding while extracting contextual signals from the neighborhood.

1) Prompt for Message Passing Aggregation: During iterative message passing, each node's representation is refined using the texts of *all neighbors*. To reduce computational overhead, nodes are processed in batches: prompts are first constructed for all nodes in a batch, and then submitted jointly to the LLM in a single evaluation call. This ensures both scalability and semantic fidelity.

The structure of each prompt instructs the LLM to integrate semantic content from the target node and its neighbors:

LLM Aggregation Prompt

You are assisting with a document classification task on a graph. Each node is a document. You are refining the representation of one document based on its own content and the content of all its neighbors.

Task: Enhance the semantic representation of the target document using its original content and the content of all neighboring documents. This enriched representation will be used in subsequent message passing steps and for final classification.

Batch Context: You are processing nodes in batch #{BATCH_ID} of {TOTAL_BATCHES}. Batching is only for computational efficiency; all neighbors of each node should be considered regardless of batch membership.

For Each Node in the Batch, Provide:

- **Original Document:** {ORIGINAL_TEXT}
- **Current Representation (from previous steps):** {CURRENT_STATE}
- **Neighbor Documents:** NEIGHBOR_TEXT_1, NEIGHBOR_TEXT_2, . . .

Output: For each node in the batch, return a semantically enriched version of the document that integrates both the original text and informative content from all neighbors. Each output should be clearly aligned with its corresponding input node.

2) Prompt for Node Classification: After message passing completes, the final node representation is provided to the LLM for classification. This prompt focuses on the enriched semantic embedding and omits batch or intermediate structural details:

LLM Node Classification Prompt

You are performing node classification in a graph of documents. Each document belongs to a topic category.

Task: Based on the enriched representation of the document (after message passing), predict the most likely topic.

Available Classes:

- sci.space
- rec.sport.hockey
- talk.politics.mideast
- misc.forsale
- ...

Final Representation of Document:
{ENRICHED_DOCUMENT_TEXT}

Output: Return only the predicted class label for this document.

3) Prompt Awareness and Semantic Traceability: Each prompt explicitly communicates the current computation step and the intended use of the output. During message passing, the LLM is informed that all neighbors should be considered, and that enriched representations will be used in subsequent iterations and for final classification. Batch indicators (e.g., {BATCH_ID}) provide computational context but do not restrict neighbor aggregation. The classification prompt then uses the final node embedding to ensure accurate label prediction.

Together, these prompt designs enable modular, interpretable, and semantically grounded reasoning over graph-structured textual data, forming the linguistic backbone of our LLM-guided message passing framework.

IV. Experiments

We organize our experiments into six components: implementation settings, baselines, datasets, results, runtime analysis, and message-passing evaluation.

A. Implementation Settings

We use an NVIDIA GeForce RTX 4090 GPU for our experiments. For all LLM-based operations, we utilize the LLaMA-3.1 model accessed via the open-source library Ollama[1]. To support graph-based reasoning, we construct a k-nearest neighbors (k-NN) document graph, where each node represents a document and is initialized with an LLM-derived embedding. Edges are formed by connecting each node to its top-k most semantically similar neighbors based on cosine similarity of the embeddings.

[1]https://ollama.com/

B. Baselines

To evaluate the effectiveness of LLM-GMP, we benchmark against a diverse set of baseline methods spanning **node classification** and **community detection**. While node classification emphasizes predicting class labels based on local features, community detection focuses on identifying clusters of nodes with dense intra-cluster connectivity and sparse inter-cluster links, often corresponding to functional roles, shared attributes, or emergent behaviors in real-world systems.

Classical community detection methods primarily rely on topology. Cut-based approaches, such as min-cut or normalized cut, recursively partition graphs to minimize edge cuts between groups. Local expansion methods, in contrast, grow communities around high-centrality seeds by iteratively optimizing community fitness measures, often aided by random walks or centrality heuristics. Although effective in some settings, these methods are limited by their neglect of attribute information and their sensitivity to sparsity and scalability challenges.

We therefore compare LLM-GMP against four major categories of baselines:

- **Random:** Assigns labels uniformly at random, serving as a lower-bound chance-level baseline.
- **Community Detection (Topology-only):** Methods that cluster nodes using only graph structure.
  - **Infomap** [18]: Models information flow via random walks and compresses these flows to reveal communities.
  - **Louvain** [1]: A widely used modularity optimization method for fast hierarchical community detection.
  - **SCD** [13]: Expands communities through nearest-greater-centrality neighbors using fuzzy relation criteria.
- **Embedding-based: DeepWalk** [17] performs random walks on the graph and treats them as sentences for a Skip-gram model, producing node embeddings that capture structural similarity in an unsupervised manner.
- **Graph Neural Networks (Supervised):** Models trained with partial label supervision.
  - **GCN** [10]: Learns node features via symmetric neighborhood aggregation with fixed weights.
  - **GAT** [22]: Incorporates self-attention to assign varying importance to neighbors, enabling anisotropic aggregation.
  - **GIN** [29]: A highly expressive GNN with discriminative power equivalent to the Weisfeiler–Lehman isomorphism test.

This selection spans topology-centric, embedding-driven, and supervised paradigms, providing a comprehensive backdrop for evaluating LLM-GMP, which in-

TABLE I
STATISTICS OF THE DATASETS USED IN EXPERIMENTS

Dataset	Nodes	Edges	Classes	Homophily
Cora	2,708	5,429	7	High
CiteSeer	3,327	4,732	6	High
PubMed	19,717	44,338	3	High
Cornell	183	295	5	Low
Texas	183	309	5	Low
Wisconsin	251	499	5	Low
Washington	229	443	5	Low
Polbooks	105	441	3	High
AdjNoun	112	425	2	Mixed

stead performs zero-shot inference through LLM-guided message passing without requiring task-specific training or labels.

C. Datasets

We evaluate LLM-GMP on a diverse set of benchmark datasets for node classification. These include citation networks, web-based graphs, and other relational networks that differ in size, density, and levels of homophily. A summary of dataset statistics is provided in Table I.

- **Citation Networks:** *Cora*, *CiteSeer*, and *PubMed* are standard benchmarks where nodes represent scientific publications and edges denote citation links [30]. The task is to classify each paper into its research area, with labels spanning 7, 6, and 3 categories, respectively. These datasets exhibit high homophily, meaning connected nodes are likely to share the same label.
- **WebKB:** *Cornell*, *Texas*, *Wisconsin*, and *Washington* are graphs constructed from computer science department web pages at different universities [16]. Nodes are individual web pages, edges represent hyperlinks, and the classification task involves 5 categories (*student*, *faculty*, *course*, *staff*, and *department*). These datasets are more heterophilic, where connected nodes often belong to different classes.
- **Other Networks:**
  - *Polbooks* is a co-purchase network of 105 political books sold on Amazon, connected by 441 edges representing frequent co-purchasing [13]. The classification task involves 3 categories: *liberal*, *conservative*, and *neutral*, and the graph is highly homophilic.
  - *AdjNoun* is a word co-occurrence network derived from the novel *David Copperfield*, where nodes are 112 nouns or adjectives and edges connect words appearing adjacently in the text [13]. The task is binary classification into *adjective* or *noun*, and the dataset exhibits mixed homophily.

D. Results

We evaluate LLM-GMP against a broad set of baselines across nine benchmark datasets, reporting node classification accuracy in Table II. The baselines include random assignment, community detection methods (*Infomap*, *Louvain*, *SCD*), embedding-based methods (*DeepWalk*), and popular GNN models (*GCN*, *GAT*, *GIN*). In contrast, LLM-GMP operates in a zero-shot setting without access to training labels.

On the homophilic citation networks (*Cora*, *CiteSeer*, *PubMed*), supervised GNNs such as GCN and GAT achieve the best results, reaching over 80% accuracy on Cora. While LLM-GMP trails these supervised methods, it still performs significantly better than community-detection or random baselines, confirming that LLM-guided message passing is able to capture non-trivial structural and semantic information even without labels. For instance, on CiteSeer, our approach achieves 61.8% accuracy, far surpassing Infomap (30.4%) or DeepWalk (46.5%).

On the heterophilic WebKB datasets (*Cornell*, *Texas*, *Wisconsin*, *Washington*), LLM-GMP shows particularly strong results. It outperforms all baselines on Cornell (61.7%) and Wisconsin (60.7%), demonstrating that iterative LLM-based aggregation is well-suited to settings where neighboring nodes often belong to different classes. Even when not the best overall, such as on Texas and Washington, our method remains competitive with strong GNN baselines, consistently surpassing traditional clustering methods. This highlights the robustness of LLM-GMP in challenging heterophilic settings where message passing often struggles.

On the additional networks, LLM-GMP achieves competitive results as well. On *Polbooks*, where the homophily level is high and several baselines perform strongly, our method achieves 76.2%, outperforming random and community-detection approaches but falling slightly behind embedding-based and supervised GNNs.

TABLE II
NODE CLASSIFICATION ACCURACY (%) OF BASELINE ALGORITHMS AND LLM-GMP (ZERO-SHOT) ACROSS NINE BENCHMARK DATASETS. THE BEST PERFORMANCE FOR EACH DATASET IS HIGHLIGHTED IN BOLD.

Dataset	**Random**	**Infomap**	**Louvain**	**SCD**	**DeepWalk**	**GCN**	**GAT**	**GIN**	**LLM-GMP**
					Citation Networks (Homophilic)				
Cora	14.80	43.90	64.70	33.50	70.00	**81.00**	80.90	70.00	61.85
CiteSeer	14.90	30.40	47.50	12.90	46.50	**70.80**	68.30	48.10	61.82
PubMed	34.80	38.90	69.60	30.60	69.90	**78.70**	76.30	75.70	58.31
					Web-KB Networks (Heterophilic)				
Cornell	29.73	29.73	32.43	43.24	37.84	43.24	37.84	37.84	**61.74**
Texas	43.24	51.35	51.35	67.57	59.46	**70.27**	64.86	37.84	66.90
Wisconsin	31.37	27.45	39.22	56.86	45.10	56.86	52.94	39.22	**60.73**
Washington	30.19	41.51	32.08	11.32	52.83	45.28	**56.60**	35.85	50.62
					Other Networks				
Polbooks	47.62	85.71	**90.48**	85.71	85.71	85.71	**90.48**	85.71	76.19
AdjNoun	56.52	52.17	43.48	69.57	69.57	69.57	73.91	82.61	**86.96**

In contrast, on the *AdjNoun* dataset, LLM-GMP achieves the best overall performance at 86.9%, surpassing even the best-performing supervised GNNs (GIN at 82.6%). This suggests that in settings with mixed or lower homophily, LLM-guided message passing can generalize more effectively than purely numerical aggregation functions.

Overall, the results show that LLM-GMP provides consistent and often substantial improvements over topology-only baselines, while approaching the performance of fully supervised GNNs in several cases. Notably, in heterophilic and mixed-homophily graphs, LLM-GMP achieves state-of-the-art performance in a zero-shot setting, demonstrating the promise of integrating LLMs into graph learning pipelines.

E. Runtime Analysis

Figure 1 reports the execution times of LLM-GMP across the nine datasets. Smaller graphs such as Polbooks (105 nodes) complete within minutes, while AdjNoun (112 nodes) requires longer runtime (13 minutes) due to higher average degree and denser neighborhoods. Medium-sized WebKB datasets (Cornell, Texas, Washington, Wisconsin) scale to 29–47 minutes, as their heterophilic structures increase message-passing complexity. Citation networks (Cora and CiteSeer) demand 81 and 97 minutes, respectively, since they contain thousands of nodes and edges. PubMed, the largest dataset (19,717 nodes, 44,338 edges), dominates runtime at 753 minutes, highlighting the computational cost of LLM-driven message passing at scale. Overall, runtime grows with both graph size and structural density, with neighborhood aggregation being the primary bottleneck. Future work may reduce this overhead via parallelization and batching strategies.

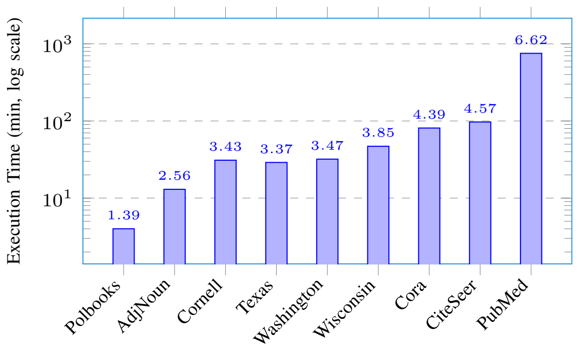

Fig. 1. Execution time (log-scaled minutes) of LLM-GMP across nine datasets, reflecting scalability with respect to graph size and density.

F. Ablation Study

To evaluate the effectiveness of structured graph message passing in LLM-GMP, we perform an ablation study by comparing it against a baseline where the LLM is used directly for node classification without our graph message passing (GMP) framework. In this baseline, the LLM receives the entire graph in textual form and attempts to classify nodes in a single step, without any message passage approach.

As shown in Figure 2, LLM-GMP consistently outperforms the LLM-only setting across all datasets. The improvement is especially pronounced on larger graphs such as *Cora* and *CiteSeer*. In addition, in large graphs such as *PubMed*, the LLM baseline fails due to context window limitations and it is not able to produce any outcome. These results highlight that decomposing reasoning into iterative, local exchanges through GMP is crucial for scalability and accuracy in zero-shot graph learning.

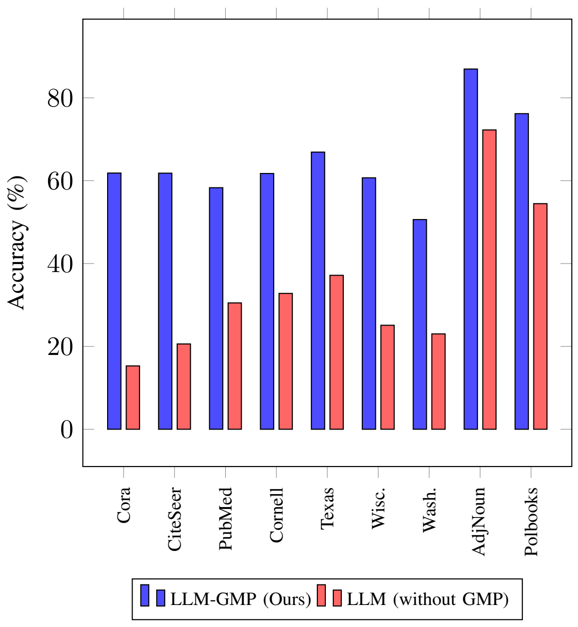

Fig. 2. Ablation study on zero-shot message passing. LLM-GMP (blue) significantly outperforms the LLM baseline without graph message passing (red), which struggles on larger graphs due to context window constraints and lack of structured propagation.

V. Conclusion

We introduced **Large Language Model Graph Message Passing** (LLM-GMP), a novel framework for zero-shot graph learning that reformulates message passing as iterative, task-aware textual communication between nodes. Unlike traditional GNNs or graph-to-text approaches, LLM-GMP enables interpretable and distributed reasoning without task-specific training, while mitigating LLM limitations such as hallucinations and context-window constraints through structured, level-wise exploration.

Extensive experiments across diverse benchmarks demonstrate that LLM-GMP consistently outperforms strong baselines and achieves accuracy competitive with advanced graph learning methods. Our ablation study further confirms that structured graph message passing is essential, as direct LLM reasoning without it performs poorly, particularly on larger graphs. Moreover, the framework scales effectively with graph size while improving efficiency over prior LLM-based approaches. These findings establish LLM-GMP as a practical and effective framework for zero-shot graph learning.

By treating nodes as lightweight reasoning agents, LLM-GMP provides a flexible, interpretable, and scalable framework for graph reasoning, naturally supporting parallel and distributed inference across large networks. This work opens new directions for integrating LLM-based inference with graph learning, advancing the development of efficient, label-free, and agentic approaches at the intersection of GNNs, LLMs, and scalable AI systems.

References

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[2] J. Chen, Y. Geng, Z. Chen, J. Z. Pan, Y. He, W. Zhang, I. Horrocks, and H. Chen. Zero-shot and few-shot learning with knowledge graphs: A comprehensive survey. *Proceedings of the IEEE*, 111(6):653–685, 2023.

[3] Z. Chen, H. Mao, H. Li, W. Jin, H. Wen, X. Wei, S. Wang, D. Yin, W. Fan, H. Liu, and J. Tang. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor. Newsl.*, 25(2):42–61, Mar. 2024.

[4] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, and Z. Duan. Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456, 2020.

[5] W. Fan, S. Wang, J. Huang, Z. Chen, Y. Song, W. Tang, H. Mao, H. Liu, X. Liu, D. Yin, et al. Graph machine learning in the era of large language models (llms). *arXiv preprint arXiv:2404.14928*, 2024.

[6] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[7] J. Guo, L. Du, H. Liu, M. Zhou, X. He, and S. Han. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.

[8] X. He. Awesome-graph-llm: A collection of awesome things about graph-related llms, 2024. Accessed: 2025-03-27.

[9] N. Ibrahim, S. Aboulela, A. Ibrahim, and R. Kashef. A survey on augmenting knowledge graphs (kgs) with large language models (llms): models, evaluation metrics, benchmarks, and challenges. *Discover Artificial Intelligence*, 4(1):76, 2024.

[10] T. Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

[12] X. Li, W. Chen, Q. Chu, H. Li, Z. Sun, R. Li, C. Qian, Y. Wei, C. Shi, Z. Liu, et al. Can large language models analyze graphs like professionals? a benchmark, datasets and models. *Advances in Neural Information Processing Systems*, 37:141045–141070, 2024.

[13] W. Luo, N. Lu, L. Ni, W. Zhu, and W. Ding. Local community detection by the nearest nodes with greater centrality. *Information Sciences*, 517:377–392, 2020.

[14] S. K. Mohamed, A. Nounu, and V. Nováček. Biological applications of knowledge graph embedding models. *Briefings in Bioinformatics*, 22(2):1679–1693, 02 2020.

[15] M. F. Naeem, Y. Xian, F. Tombari, and Z. Akata. Learning graph embeddings for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 953–962, June 2021.

[16] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[17] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[18] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008.

[19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[20] K. e. a. Shen. Grapheval36k: A benchmark for evaluating graph understanding in llms. *Proceedings of CVPR*, 2024.

[21] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500, 2024.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[23] D. Wang, Y. Zuo, F. Li, and J. Wu. Llms as zero-shot graph learners: Alignment of gnn representations with llm token embeddings. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 5950–5973. Curran Associates, Inc., 2024.

[24] S. Wang, J. Huang, Z. Chen, Y. Song, W. Tang, H. Mao, W. Fan, H. Liu, X. Liu, D. Yin, and Q. Li. Graph machine learning in the era of large language models (llms). *ACM Trans. Intell. Syst. Technol.*, May 2025. Just Accepted.

[25] X. Wang, J. Li, L. Yang, and H. Mi. Unsupervised learning for community detection in attributed networks based on graph convolutional network. *Neurocomputing*, 456:147–155, 2021.

[26] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), Dec. 2022.

[27] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[28] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[29] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[30] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377, 2019.

[31] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.