# Investigating the Limits of Graph Foundation Model in Real-World Travel Recommendation Systems

Nayoung Lee[1], Gunmin Lee[2], and Donghun Lee[1]

[1] Department of Mathematics, Korea University, Seoul 02841, Korea
{nazero, holy}@korea.ac.kr
[2] Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea
gunmin.lee@rllab.snu.ac.kr

**Abstract.** Graph foundation models (GFMs) have demonstrated remarkable potential in capturing intricate relational patterns, achieving state-of-the-art results in numerous graph-centric tasks. However, their real-world applicability remains underexplored in highly domain-specific contexts, such as travel recommendation. In this paper, we present a comprehensive evaluation of GFMs for large-scale travel recommendation tasks using a bipartite user–destination dataset of 86,761 travelers within South Korea. We compare representative GFM against both conventional graph-based methods and vector-based methods. Contrary to the prevailing expectation that GFMs should outperform traditional architectures, our empirical findings reveal that domain-specific constraints can dilute the benefits of extensive multi-hop message passing, leading to suboptimal performance. Our work highlights a critical need to validate GFMs against domain-specific constraints, offering a roadmap for their future adaptation and optimization in real-world applications.

**Keywords:** Graph foundation model · Travel recommendation · Domain-specific constraints.

## 1 Introduction

Graph neural networks (GNNs)[15] have emerged as a powerful approach for making inferences and predictions on graph-structured data, offering a natural way to process relational information. However, these models often get over-specialized to specific datasets[16], limiting their generalization capabilities across different domains and tasks. To address this limitation, graph foundation models (GFMs) which show better generalization and capabilities in various graph-related tasks, were introduced[13]. These models, trained on diverse graph data, capture universal graph patterns that can be transferred across different domains and applications.

In this work, we test these claims by applying GFM to travel recommendation task. This domain presents a unique challenge where the data is naturally represented as a collection of subgraphs, each corresponding to a real

travel trajectory [18]. While this structure might seem ideal for graph-based approaches [3], our comprehensive experiments reveal unexpected limitations of both GNNs and GFM in this context. Surprisingly, our results show that simpler vector-based approaches outperform sophisticated graph-based methods, including GFM. Through careful analysis, we identify two key findings that contribute to the broader understanding of GFM in real world applications:

1. We identify fundamental limitations of GFM in domain-specific applications, particularly in travel recommendation systems. Our analysis reveals that indirect connections in user-location-user patterns act as noise rather than meaningful relationships, demonstrating how domain characteristics can fundamentally limit GFM effectiveness.
2. We empirically validate these limitations through comprehensive experiments with various graph-based approaches. Our results show that traditional GFM performance metrics like hop count and dataset size can inversely correlate with recommendation quality, challenging conventional assumptions about GFM applications in specific domains.

## 2 Problem Setting

Travel recommendation problem is a task that recommends new users tours based on data of previous users. This problem has advantages over other recommendation tasks when utilizing graph based datasets [4]. We focus on developing a personalized recommendation system for domestic travel destinations in South Korea. We aim to recommend top-$k$ destinations based on user characteristics, including personal preferences, gender, and age group. Furthermore, we extend our approach to address the generalized problem that can be applied to various recommendation problems.

### 2.1 Dataset Characteristics

As a concrete case study for our approach, we utilize a dataset consisting of travel records of 86,761 users during August to November 2022 who have taken domestic trips in South Korea. The data naturally forms a bipartite graph structure [1], where:

- Users ($\mathbb{U}$) and destinations ($\mathbb{D}$) form two distinct node sets.
- Edges ($\mathbb{E}$) represent visits from users to destinations.

This structure suggests natural affinity for graph-based approaches, as it creates a network of connections between users through their shared destination visits. Following common practice in recommendation systems, these shared destinations could theoretically indicate similarity in personal preferences.

### 2.2 Problem Formulation

Formally, our input for the travel recommendation problem can be defined as follows:

$$\mathbb{U} = \{u_1, u_2, \ldots, u_n\},$$

$$\mathbb{D} = \{d_1, d_2, \ldots, d_m\},$$

$$V \in \{0,1\}^{n \times m}, \quad \text{where } v_{ij} = \begin{cases} 1 & \text{if user } i \text{ visited destination } j, \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathbb{U}$ is a set of $n$ users with their features $X_u$ including age group, gender, and personal preferences, $\mathbb{D}$ is a set of $m$ destinations, and $\mathbb{V}$ which is a historical visit matrix.

The goal of the problem is to recommend a ranked list of top-$k$ destinations for any given user $u_q$ that best matches personal preferences. We define these "best matches" through a scoring function $f(u_q, d_j)$ that quantifies how well destination $d_j \in \mathbb{D}$ aligns with user preferences, incorporating both demographic attributes and implicit features from past interactions. The function

$$f : \mathbb{U} \times \mathbb{D} \to \mathbb{R}$$

outputs a score indicating the predicted utility of each destination for the user. The recommended destinations are then selected by ranking all destinations based on these scores and choosing the top-$k$:

$$\{d_1^*, d_2^*, \ldots, d_k^*\} = \arg\operatorname*{top}_{d \in \mathbb{D}} k \; f(u_q, d).$$

To evaluate the recommendation results, we employ five evaluation metrics:

- **Two error-based metrics**: mean squared error (MSE) and mean absolute error (MAE) to measure the accuracy of predictions.
- **One similarity metric**: cosine similarity (CS) to assess the semantic closeness of recommendations.
- **Two ranking metrics**: precision@$k$ and recall@$k$ to evaluate the quality of top-$k$ recommendations.

## 3 Methods

We compare three distinct approaches for travel recommendation: GFM, graph-based models, and vector-based models. The comparison of the three different approaches is shown in table 1.

| | GFM | Graph-based | Vector-based |
|---|---|---|---|
| **Info. Flow** | Unrestricted multi-hop | Limited neighborhood | Direct mapping |
| **Relations** | All possible paths | Local connections | Point-to-point mappings |
| **Complexity** | Exponential growth | Linear growth | Constant |
| **Example** | [17] | [11], [7], [6] | [10], [8], [2] |

**Table 1.** Comparison of models by information propagation

### 3.1 Graph Foundation Model

The Graph Foundation Model (GFM) approach adapts GraphAny to our travel recommendation task, focusing on the bipartite structure of user-place relationships. The model processes information through dual channels: a low-pass channel and a high-pass channel. The low-pass channel captures smooth, global patterns across the graph by aggregating neighborhood information, while the high-pass channel emphasizes local variations and structural differences between nodes. These channels are defined as:

$$L_k = D^{-1}AX \quad \text{and} \quad H_k = (I - D^{-1}A)X, \tag{1}$$

where $L_k$ represents the low-pass channel output, $H_k$ denotes the high-pass channel output, $D$ is the degree matrix, $A$ is the adjacency matrix, $I$ is the identity matrix, and $X$ represents the node feature matrix. The low-pass channel $L_k$ implements a normalized adjacency operation, effectively averaging features across connected nodes, while the high-pass channel $H_k$ captures the deviation of each node's features from its neighborhood average.

This structure allows for multi-hop information propagation across the graph, though this can potentially lead to noise accumulation through indirect relationships. The model's learning objective is formulated through a combination of reconstruction and link prediction [12] losses:

$$\mathcal{L} = \|X - \hat{X}\|^2 + \lambda(-\sum_{(i,j)}[y_{ij}\log(\hat{y}_{ij}) + (1 - y_{ij})\log(1 - \hat{y}_{ij})]), \tag{2}$$

where $|X - \hat{X}|^2$ represents the reconstruction loss measuring the model's ability to preserve node features, $y_{ij}$ indicates whether nodes $i$ and $j$ are connected in the original graph, $\hat{y}_{ij}$ is the model's predicted probability of connection between nodes $i$ and $j$, and $\lambda$ is a balancing hyperparameter. The reconstruction term ensures feature preservation, while the binary cross-entropy term guides the model to accurately predict graph structure.

### 3.2 Graph-based Models

Graph-based models provide a more controlled approach to utilizing network structure. Each graph based model implements distinct strategies for managing information propagation and neighborhood aggregation:

- GCN[11] employ a layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \tag{3}$$

where $\tilde{A} = A + I$ is the adjacency matrix with self-loops, $\tilde{D}$ is the corresponding degree matrix, $H^{(l)}$ represents node features at layer $l$, $W^{(l)}$ is the learnable weight matrix, and $\sigma$ is a non-linear activation function.

- GraphSAGE[7] employs a neighborhood sampling strategy:

$$h_v^{(l+1)} = \sigma(W^{(l)} \cdot \text{AGGREGATE}(\{h_u^{(l)}, \forall u \in \mathcal{N}(v)\})), \tag{4}$$

where $\mathcal{N}(v)$ is a sampled set of neighbors for node $v$, and AGGREGATE is a permutation-invariant aggregation function.

- Node2Vec[6] learns embeddings through biased random walks, balancing between breadth-first and depth-first graph exploration:

$$f : V \to \mathbb{R}^d, \tag{5}$$

where $f$ is the embedding function mapping nodes to a d-dimensional space, optimized to preserve both local and global network structure.

These methods are trained using binary cross-entropy loss for link prediction:

$$\mathcal{L} = -\sum_{(u,p)} [y_{up} \log(\hat{y}_{up}) + (1 - y_{up}) \log(1 - \hat{y}_{up})], \tag{6}$$

where $(u, p)$ represents a user-place pair, $y_{up}$ indicates whether user $u$ has visited place $p$, and $\hat{y}_{up}$ is the model's predicted probability of user $u$ visiting place $p$. The loss function encourages the model to accurately predict existing connections while avoiding false positives in the user-place interaction graph.

Unlike GFM's propagation, these graph-based methods provide more controlled information flow, potentially reducing noise from indirect relationships while retaining essential structural information in the user-place interaction network. Moreover, whereas equation (2) focuses on user-place features or other scalar-based predictions, equation (6) distinctly zeroes in on explicit $(u, p)$ interactions via a binary cross-entropy loss. This emphasis on direct user-place link prediction, whether user $u$ visits place $p$, offers a clearer objective for learning these interactions compared to the more generalized or feature-based loss used in equation (2).

### 3.3 Vector-based Models

Vector-based methods take the most direct approach, focusing solely on feature relationships without utilizing graph structure. These methods represent users through demographic information and explicit personal preferences, while places are represented by their location attributes and characteristics. The models employ different architectures:

Autoencoder model uses a deterministic encoder-decoder structure with a bottleneck layer:

$$z = f_\phi(X) \quad \text{and} \quad \hat{X} = g_\theta(z), \tag{7}$$

which is trained with MSE reconstruction loss:

$$\mathcal{L} = \|X - \hat{X}\|^2. \tag{8}$$

VAE variants (VAE and BetaVAE) introduce probabilistic encoding through a variational inference framework:

$$q_\phi(z|X) = \mathcal{N}(\mu_\phi(X), \sigma_\phi(X)) \tag{9}$$

with a combined loss function:

$$\mathcal{L} = \|X - \hat{X}\|^2 + \beta \cdot KL(q_\phi(z|X)\|p(z)), \tag{10}$$

where $\beta = 1$ for VAE and $\beta > 1$ for BetaVAE. These methods serve as an important baseline, demonstrating the effectiveness of direct feature relationships without the complexity of graph structure.

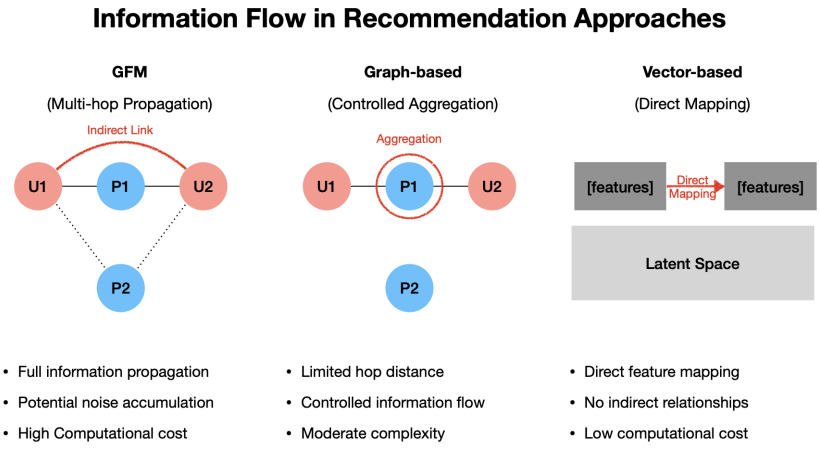


**Fig. 1.** Information flow analysis in different recommendation approaches

## 4 Results

In this section, we present a comprehensive analysis of our experimental results, comparing various models across different metrics and settings. We evaluate the performance using MSE, MAE, CS, precision@$k$, and recall@$k$ for different values of $k$ [9].

### 4.1 Overall Performance Comparison

Table 2 presents overall evaluation results for all models at $k$=5, which show how the performance compares depending on the methods and the models.

**Table 2.** Model performance comparison ($k$=5)

| Method | Model | MSE (↓) | MAE (↓) | CS (↑) | P@5 (↑) | R@5 (↑) |
|---|---|---|---|---|---|---|
| GFM | GraphAny | 20.9747 | 2.3957 | 0.9716 | 0.0008 | 0.0013 |
| Graph-based | GCN | 13.2601 | 1.9900 | 0.9813 | 0.0128 | 0.0211 |
| | GraphSAGE | 16.0332 | 2.4205 | 0.9674 | 0.0001 | 0.0002 |
| | Node2Vec | 20.1005 | 2.3473 | 0.9726 | 0.0009 | 0.0016 |
| Vector-based | VAE | 8.4498 | 1.7581 | 0.9826 | 0.0022 | 0.0037 |
| | β-VAE | 13.5527 | 2.1140 | 0.9763 | 0.0012 | 0.0022 |
| | AE | **1.0950** | **0.7028** | **0.9938** | **0.0331** | **0.0592** |

From these results, we observe that the vector-based methods outperform the graph-based methods on most metrics, with AE achieving the best overall performance in terms of all five metrics. This indicates that, in the context of travel recommendations, simple vector representations may provide more reliable embeddings than graph-based approaches. GraphAny, GraphSAGE, and Node2Vec demonstrate comparatively lower ranking performance, suggesting challenges in effectively capturing indirect relationships while minimizing noise [14]. Meanwhile, GCN shows moderate performance, indicating that some localized graph information can be beneficial, although it is still outperformed by AE.

### 4.2 Impact of Top-$k$ Selection

To understand how different models perform with varying numbers of recommendations, $k$, we analyze each model's performance across three different $k$ values, $k = 5, 10$, and 20. Table 3 shows the precision and recall metrics for different models, where the number of recommendations is set to 5, 10, and 20.

P The results show that the precision and recall metrics can vary significantly with $k$. The differential behavior of graph-based and vector-based models as k increases provides insight into how architectural choices affect recommendation quality. GCN exhibits a gradual precision degradation with linear recall growth, suggesting GCN is capturing increasingly distant relationships through indirect user-location-user connections. In contrast, vector-based models, particularly AE, demonstrate superior precision performance, accompanied by diminishing recall growth. While graph-based models continue to find recommendations through indirect paths, potentially incorporating noise from users who visited the same locations for different purposes, vector-based models strictly adhere to direct similarity measures, leading to better precision at lower k values but steeper performance drops when exhausting highly similar recommendations.

**Table 3.** Precision and recall performance for top-5, top-10, and top-20 similar trips

| Method | Model | P@5 (↑) | P@10 (↑) | P@20 (↑) | R@5 (↑) | R@10 (↑) | R@20 (↑) |
|---|---|---|---|---|---|---|---|
| GFM | GraphAny | 0.0008 | 0.0008 | 0.0008 | 0.0013 | 0.0026 | 0.0057 |
| Graph-based | GCN | 0.0128 | 0.0121 | **0.0118** | 0.0211 | 0.0385 | **0.0735** |
| | GraphSAGE | 0.0001 | 0.0001 | 0.0001 | 0.0002 | 0.0006 | 0.0011 |
| | Node2Vec | 0.0009 | 0.0008 | 0.0008 | 0.0016 | 0.0028 | 0.0058 |
| Vector-based | VAE | 0.0022 | 0.0022 | 0.0019 | 0.0037 | 0.0073 | 0.0127 |
| | β-VAE | 0.0012 | 0.0013 | 0.0012 | 0.0022 | 0.0042 | 0.0081 |
| | AE | **0.0331** | **0.0180** | 0.0100 | **0.0592** | **0.0641** | 0.0713 |

### 4.3 Dataset Size Analysis

We conducted experiments with three different dataset sizes, $|\mathbb{V}|$, across multiple models to analyze the impact of dataset size on model performance. Table 4 presents these results.

**Table 4.** Model performance across different dataset size ($k = 5$)

| Model | $\lvert\mathbb{V}\rvert$ | MSE (↓) | MAE (↓) | P@5 (↑) | R@5 (↑) |
|---|---|---|---|---|---|
| GraphAny | 2000 | 20.7186 | 2.3870 | 0.0089 | 0.0159 |
| | 5000 | 21.0153 | 2.4150 | 0.0034 | 0.0066 |
| | 86761 | 20.9747 | 2.3957 | 0.0008 | 0.0013 |
| GCN | 2000 | 16.1780 | 2.1589 | 0.0292 | 0.0551 |
| | 5000 | 15.0940 | 2.1143 | 0.0183 | 0.0378 |
| | 86761 | 13.2601 | 1.9900 | 0.0128 | 0.0211 |
| GraphSAGE | 2000 | 28.5383 | 2.7780 | 0.0009 | 0.0019 |
| | 5000 | 18.1254 | 2.4444 | 0.0007 | 0.0017 |
| | 86761 | 16.0332 | 2.4205 | 0.0001 | 0.0002 |
| Node2Vec | 2000 | 22.0694 | 2.4124 | 0.0109 | 0.0196 |
| | 5000 | 22.1293 | 2.4453 | 0.0028 | 0.0059 |
| | 86761 | 20.1005 | 2.3473 | 0.0009 | 0.0016 |
| VAE | 2000 | 8.4498 | 1.7581 | 0.0022 | 0.0037 |
| | 5000 | 6.6342 | 1.6820 | 0.0081 | 0.0161 |
| | 86761 | 6.6815 | 1.6831 | 0.0186 | 0.0367 |
| β-VAE | 2000 | 13.5527 | 2.1140 | 0.0012 | 0.0022 |
| | 5000 | 10.0136 | 1.9200 | 0.0070 | 0.0141 |
| | 86761 | 9.3613 | 1.8823 | 0.0123 | 0.0233 |
| AE | 2000 | **1.0950** | **0.7028** | **0.0331** | **0.0592** |
| | 5000 | **2.1770** | **1.0420** | **0.0204** | **0.0412** |
| | 86761 | **2.5165** | **1.1164** | **0.0243** | **0.0478** |

Interestingly, we observe that increasing the dataset size does not necessarily lead to better performance across all models. While GFM and graph-based models generally show modest improvements in MSE and MAE metrics with larger datasets, P@5 and R@5 values decrease.

### 4.4 Impact of Information Propagation Range

To better understand how the range of information propagation affects model performance, we conducted experiments with different hop counts for GCN, and the walk length and context size for Node2Vec. Note that we focus on these two models as they show different mechanisms of information propagation through graph structure, while other models either do not use graph propagation or have fixed propagation patterns. The results are shown in Table 5.

**Table 5.** Performance comparison across different information propagation ranges

| **Model** | **Range** | **MSE** (↓) | **MAE** (↓) | **P@5** (↑) | **R@5** (↑) |
|---|---|---|---|---|---|
| GCN | 1-hop | 16.1004 | 2.2242 | **0.0443** | **0.0927** |
| | 2-hop | **14.9774** | **2.1022** | 0.0190 | 0.0373 |
| | 3-hop | 15.2528 | 2.1256 | 0.0118 | 0.0247 |
| Node2Vec | Walk-5/Context-3 | **19.5810** | **2.3366** | **0.0148** | **0.0294** |
| | Walk-10/Context-5 | 23.2161 | 2.4858 | 0.0021 | 0.0046 |
| | Walk-20/Context-10 | 22.4065 | 2.4805 | 0.0016 | 0.0032 |

The results show that increasing the propagation range leads to a decrease in recommendation accuracy. For GCN, 2-hop propagation achieves optimal MSE and MAE, but precision and recall metrics consistently decrease as the hop count increases. Node2Vec shows similar behavior, with the shortest range achieving the best performance across all metrics. These findings suggest that while broader information propagation helps minimize overall prediction error, it lowers the model's ability to make precise recommendations.

**Discussion** The experimental results give three key insights about embedding approaches in travel recommendation systems.

First, the superior performance of vector-based methods across metrics challenges a fundamental assumption in recommendation systems that edges in graph structures lead to better recommendations. While the consistently lower error rates of AE suggest that user-location interactions may be effectively captured through direct mappings, it is important to note that this advantage might be partly explained by the training objective of the AE. Specifically, AE-based models minimize a reconstruction error that is closely tied to the evaluation metrics, whereas graph-based models do not directly optimize for the same objective. Therefore, while the performance gap highlights the potential advantages

of more direct user-location representations, it also underscores the need to interpret these metrics in light of the models' respective training objectives.

Second, the degradation of graph-based models' performance with increased dataset size and propagation range reveals a structural limitation. The declining precision metrics of GCN and Node2Vec with extended propagation ranges indicate that indirect user-location-user connections may introduce more noise than signal, as users might visit the same locations for different purposes. This observation is supported by GraphSAGE's performance, where broader information sampling appears to amplify this noise rather than extract meaningful patterns.

Third, the inverse relationship between dataset size and recommendation accuracy in graph-based models, contrasted with the stable performance of vector-based approaches, suggests that the complexity of graph structures may not scale effectively with travel data. This pattern indicates that in the travel domain, the quality and directness of user-location relationships may be more crucial than the quantity of indirect connections captured through graph structures. These findings align with the notion that GNN's message-passing mechanisms may become increasingly susceptible to structural noise as graph size grows [5].

## 5 Conclusion

This study demonstrates that vector-based models can outperform graph-based models in travel recommendation systems, with AE achieving the best MSE of 1.0950. These results suggest that the inherent characteristics of travel data may need additional refinement for graph-structured representations. This study also demonstrated the need for domain-aware loss engineering and GFM architectures that better align with bipartite graphs with the specific demands of travel recommendation problems.

Moreover, this study empirically shows several limitations of GFM. Our experiments show that increasing the number of connections can degrade key metrics like precision and recall. These observations underscore that the benefits of graph-based representations in recommendation systems are highly dependent upon the domain characteristics and the effectiveness of the graph structure in capturing meaningful user-item relationships.

Future work should focus on developing hybrid architectures that can selectively use graph structures while maintaining the effectiveness of direct user-location mappings. Additionally, investigating methods for filtering meaningful connections in travel graphs and developing domain-specific message-passing mechanisms could address the current limitations of graph-based approaches in travel recommendation systems.

**Acknowledgments.** We would like to acknowledge Bosung Jung, and Gyumin Lee (Nara information) for their valuable contributions to the implementation of the VAE model and the construction of the dataset. Their insights and technical expertise significantly enhanced the quality of this work.

## References


1. Asratian, A.S., Denley, T.M., Häggkvist, R.: Bipartite graphs and their applications, vol. 131. Cambridge university press (1998)
2. Bank, D., Koenigstein, N., Giryes, R.: Autoencoders. Machine learning for data science handbook: data mining and knowledge discovery handbook pp. 353–374 (2023)
3. Chen, L., Cao, J., Liang, W., Ye, Q.: Geography-aware heterogeneous graph contrastive learning for travel recommendation. ACM Trans. Spatial Algorithms Syst. **10**(3), 27:1–27:22 (2024)
4. Chen, L., Cao, J., Wang, Y., Liang, W., Zhu, G.: Multi-view graph attention network for travel recommendation. Expert Systems with Applications **191**, 116234 (2022)
5. Dai, E., Jin, W., Liu, H., Wang, S.: Towards robust graph neural networks for noisy graphs with sparse labels. CoRR **abs/2201.00232** (2022)
6. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. pp. 855–864. ACM (2016)
7. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 1024–1034 (2017)
8. Higgins, I., Matthey, L., Pal, A., Burgess, C.P., Glorot, X., Botvinick, M.M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017)
9. Jadon, A., Patil, A.: A comprehensive survey of evaluation techniques for recommendation systems. CoRR **abs/2312.16015** (2023)
10. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR **abs/1609.02907** (2016)
12. Lichtenwalter, R., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: Rao, B., Krishnapuram, B., Tomkins, A., Yang, Q. (eds.) Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010. pp. 243–252. ACM (2010)
13. Liu, J., Yang, C., Lu, Z., Chen, J., Li, Y., Zhang, M., Bai, T., Fang, Y., Sun, L., Yu, P.S., Shi, C.: Towards graph foundation models: A survey and beyond. CoRR **abs/2310.11829** (2023)
14. Mostafa, H., Nassar, M., Majumdar, S.: On local aggregation in heterophilic graphs. CoRR **abs/2106.03213** (2021)
15. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Networks **20**(1), 61–80 (2009)

16. Wu, S., Zhang, W., Sun, F., Cui, B.: Graph neural networks in recommender systems: A survey. CoRR **abs/2011.02260** (2020)
17. Zhao, J., Zhu, Z., Galkin, M., Mostafa, H., Bronstein, M., Tang, J.: Fully-inductive node classification on arbitrary graphs. In: International Conference on Learning Representations (ICLR) (2025)
18. Zheng, Y.: Trajectory data mining: An overview. ACM Trans. Intell. Syst. Technol. **6**(3), 29:1–29:41 (2015)