AMF-GR: Adaptive Matrix Factorization and Graph Fusion for Android Library Recommendation

Abhinav Jamwal
Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee, India
abhinav_j@cs.iitr.ac.in

Sandeep Kumar
Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee, India
sandeep.garg@cs.iitr.ac.in

***Abstract*—The Android app ecosystem is heavily relying on third-party libraries (TPLs), which help developers accelerate app development and add new features efficiently. However, identifying suitable libraries from a vast and constantly changing ecosystem remains a difficult challenge. While matrix factorization (MF) based models effectively capture direct app-library interactions, they overlook higher-order relations among apps and libraries. Graph-based approaches can model these relations but often rely on fixed aggregation strategies, which limit flexibility and cause unstable training. To address this gap, we propose AMF-GR, an Adaptive Matrix Factorization and Graph Fusion framework to recommend third-party libraries in Android applications. AMF-GR combines MF embeddings with high-order graph propagation and introduces learnable fusion gates and adaptive propagation strengths to balance information from both components during training. We evaluated the proposed approach on a real-world data set that contains 31,432 Android applications and 752 distinct libraries. The results show that AMF-GR achieves higher recommendation accuracy and more stable convergence compared to traditional MF and graph-based baselines, demonstrating its effectiveness for reliable library recommendation in Android app development.**

***Index Terms*—Third-party library Recommendation, Hypergraph Neural Network, Graph-Based Recommendation, Machine Learning.**

I. Introduction

Mobile apps are now one of the main ways people use software. Recent data shows that the Google Play Store offers over 2.26 million Android apps[1]. This fast-growing market creates exciting opportunities, but it also leads to tough competition as many apps offer similar features. To shorten development cycles, improve reliability, and enhance productivity, developers frequently integrate third-party libraries (TPLs) that provide ready-to-use features and stable APIs.

With the continuous growth of the mobile ecosystem, thousands of TPLs are now available for Android developers. Although this abundance improves software reuse, it also creates a new challenge: identifying and selecting the most suitable libraries among numerous alternatives. Developers may not be aware of new or relevant TPLs, leading to redundant reimplementation or missed optimization opportunities. Therefore, it is essential to provide automated approaches that can effectively recommend useful libraries for Android app development.

Several studies have explored the recommendation of TPL for mobile or open-source projects. Early works relied on association rule mining to identify co-usage relationships among libraries, but their performance largely depends on the frequency of library co-occurrence [1]. Later, collaborative filtering (CF) and matrix factorization (MF) techniques were applied to capture app–library interactions from historical data, improving personalization but still restricted to low-order relationships [2], [3]. More recent graph-based models introduced high-order structural learning [4], [5], but most employ fixed aggregation strategies, which limits adaptability and leads to unstable optimization.

To address these limitations, we propose AMF-GR, an Adaptive Matrix Factorization and Graph Fusion framework to recommend third-party libraries in Android applications. AMF-GR integrates MF embeddings with high-order graph propagation and employs learnable fusion gates and adaptive propagation strengths to regulate the contribution of each component during training. This design enables the model to dynamically balance low and high-order collaborative signals, resulting in improved stability and recommendation accuracy.

We evaluated the proposed approach on a large-scale data set that contains 31,432 Android applications, 752 distinct libraries, and more than 500,000 app-library usage records. The results demonstrate that AMF-GR improves both accuracy and stability compared to traditional MF and graph-based baselines.

The main contributions of this paper are as follows

- We introduce AMF-GR, a hybrid recommendation framework that combines matrix factorization and graph propagation for third-party library recommendation in Android apps.
- We design an adaptive fusion mechanism that dynamically balances low- and high-order collaborative signals to enhance accuracy and training stability.
- We conducted comprehensive experiments on a large-scale Android app dataset to validate the effectiveness of the proposed approach.

[1]https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/

II. Related Work

A. Third-Party Library Recommendation

In recent years, numerous studies have investigated techniques to help developers select suitable third-party libraries (TPLs) for software and mobile app projects. Early research relied on association rule mining to discover co-usage patterns among libraries. For example, Thung et al. [1] proposed LibRec, which recommends entire libraries based on co-occurrence relationships mined from existing applications. Although such methods improve reuse awareness, their accuracy heavily depends on the frequency of library co-occurrence within projects.

Subsequent approaches used collaborative filtering (CF) to identify similarities between applications and libraries. Nguyen et al. [2] introduced CrossRec, which models project similarity by combining CF and weighting mechanisms to recommend libraries in open-source projects. He et al. [3] presented LibSeek, a matrix factorization (MF)-based method that incorporates adaptive weighting to improve diversity in mobile app recommendations. These CF- and MF-based methods have demonstrated good scalability and personalization but remain limited to low-order interactions derived from two-dimensional app–library matrices.

More recently, graph-based models have been proposed to capture high-order dependency patterns. Li et al. [6] introduced GRec, which represents apps and libraries as nodes in a bipartite graph and applies a graph neural network (GNN) to extract high-order structural information. Inspired by LightGCN [5], other methods further simplified graph aggregation and extended the approach to mobile app ecosystems. However, most graph-based solutions employ fixed propagation strategies that restrict adaptability and often lead to unstable training behavior.

Our work differs from these studies by integrating MF-based personalization with high-order graph representation learning. The proposed AMF-GR introduces learnable fusion gates and adaptive propagation strengths to dynamically balance low-order and high-order collaborative signals during training, improving both recommendation accuracy and model stability.

B. Graph Representation Learning for Recommendation

Graph representation learning has achieved great success in the various recommendation contexts by effectively modeling complex dependencies and nonlinear interactions between different entities. Recent advances in simplifying the graph convolution architectures, such as those reported in [5], have shown that heavy transformation and activation components can be removed to considerably improve efficiency and interpretability without any loss in expressiveness. Recent works in this paradigm have explored graph contrastive learning and hypergraph propagation to enhance node representations [7], [8].

Unlike these graph-centric models, the proposed approach combine strengths of the matrix factorization and high-order graph propagation without relying purely on deep graph architectures. This hybrid design allows AMF-GR to retain the personalization ability of MF while leveraging structural information from multi-hop graph connections. Consequently, AMF-GR achieves a balance between the accuracy of the recommendation and the stability of the training, making it suitable for large-scale Android library recommendation tasks.

III. Proposed Approach

AMF-GR aims to recommend appropriate third-party libraries for Android apps by combining basic collaborative usage patterns with more complex relationships from app library interactions. To support reproducibility and further study, we have publicly released the AMF-GR implementation [2] and the MALib dataset [3], which we used in our experiments. The framework has four main phases that align with the architecture shown in Figure 1.

A. Phase 1: Interaction-Based Hypergraph Construction

We begin with a binary interaction matrix $R \in \{0,1\}^{|A|\times|L|}$ where $R_{ij} = 1$ indicates that app A_i uses library L_j.

To characterize similarity among apps and among libraries, we compute pairwise Jaccard similarity over their usage sets, a measure that has been shown to perform well in sparse interaction settings [9].

$$\mathrm{Sim}_A(i,j) = \frac{|L(A_i) \cap L(A_j)|}{|L(A_i)| + |L(A_j)| - |L(A_i) \cap L(A_j)|},$$

$$\mathrm{Sim}_L(p,q) = \frac{|A(L_p) \cap A(L_q)|}{|A(L_p)| + |A(L_q)| - |A(L_p) \cap A(L_q)|}.$$

For each app A_i, we select its similar top-apps k and form a hyperedge; the same process is applied to libraries. The resulting incidence matrices for the app and library hypergraphs are:

$$H_a(i,e) = \begin{cases} \mathrm{Sim}_A(i,j) & \text{if app } A_i \text{ belongs to hyperedge } e_j, \\ 0 & \text{otherwise,} \end{cases}$$

$$H_l(p,e) = \begin{cases} \mathrm{Sim}_L(p,q) & \text{if library } L_p \text{ belongs to hyperedge } e_q, \\ 0 & \text{otherwise.} \end{cases}$$

To normalize hypergraph propagation, we compute the node and hyperedge degree matrices:

$$D_v(i,i) = \sum_e H(i,e), \qquad D_e(e,e) = \sum_i H(i,e).$$

The normalized hypergraph propagation operator is the following:

$$\hat{H} = D_v^{-1} H D_e^{-1}.$$

[2]AMF-GR https://github.com/ABhiIITRoorkee/AMF-GR

[3]MALib data set is available at https://github.com/malibdata/MALib-Dataset.git

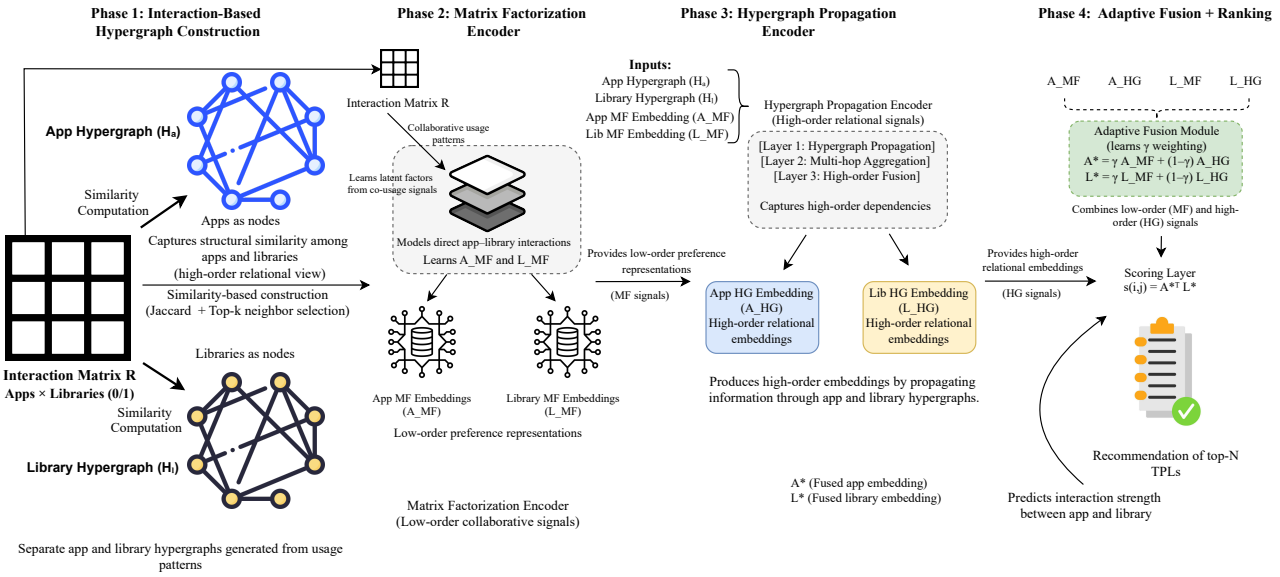

Fig. 1. Overview of the proposed AMF-GR framework. Phase 1 constructs separate app and library hypergraphs from the binary interaction matrix by computing similarity among apps and libraries and forming hyperedges based on Jaccard and top-k neighbors. Phase 2 applies matrix factorization to the same interaction matrix to learn low-order preference representations that reflect direct co-usage patterns between apps and libraries. Phase 3 propagates information through the two hypergraphs to capture high-order relational dependencies and produce hypergraph-based embeddings for apps and libraries. Phase 4 adaptively fuses the low-order and high-order representations using a learnable weighting mechanism, and the resulting fused embeddings are scored through an inner-product layer to generate the final Top-N library recommendations.

B. Phase 2: Matrix Factorization Encoder

While hypergraphs encode structural relations, they do not capture direct app–library co-usage signals. To learn these low-order preferences, we apply matrix factorization to R. Let $U \in \mathbb{R}^{|A|\times d}$ and $V \in \mathbb{R}^{|L|\times d}$ denote app and library latent factors. The MF objective is as follows:

$$\min_{U,V} \sum_{(i,j):R_{ij}=1} -\log \sigma(U_i^\top V_j) + \lambda(\|U\|^2 + \|V\|^2),$$

which corresponds to the pairwise BPR loss used during training.

This produces low-order embeddings:

$$A_{\mathrm{MF}} = U, \qquad L_{\mathrm{MF}} = V.$$

C. Phase 3: Hypergraph Propagation Encoder

To capture high-order dependencies, AMF-GR performs propagation on the app and library hypergraphs. For node features X (initialized as MF embeddings), one step of hypergraph propagation is:

$$X^{(l+1)} = \hat{H}\hat{H}^\top X^{(l)}.$$

For the app and library hypergraphs:

$$A_{\mathrm{HG}} = \frac{1}{m+1}\sum_{l=0}^{m} X_a^{(l)}, \qquad L_{\mathrm{HG}} = \frac{1}{m+1}\sum_{l=0}^{m} X_l^{(l)},$$

where m is the number of propagation layers.

This produces high-order relational embeddings that model multi-hop similarity and shared neighborhood structure.

D. Phase 4: Adaptive Fusion and Ranking

AMF-GR combines low- and high-order representations using a learnable fusion weight $\alpha \in [0, 1]$. In our design, α is a *global scalar* shared across all apps and libraries, which keeps the fusion scheme simple and prevents overfitting in sparse regions of the data. The parameter is updated jointly with all model parameters through backpropagation, and is constrained to the interval $[0, 1]$ using a sigmoid projection during training. We use the same α for both the app and library branches, ensuring that the balance between MF-based and hypergraph-based signals remains consistent across the two embedding spaces.

The fused representations are:

$$A = \alpha A_{\mathrm{MF}} + (1-\alpha)A_{\mathrm{HG}}, \qquad L = \alpha L_{\mathrm{MF}} + (1-\alpha)L_{\mathrm{HG}}.$$

For a target app i and a candidate library j, the interaction score is computed as:

$$s(i,j) = A_i^\top L_j.$$

Libraries for each app are then ranked in descending order of $s(i,j)$, and the top-N items form the final recommendations.

In these phases, the proposed approach generates expressive representations of raw usage data by capturing direct app-library co-use signals, as well as the high-order relational patterns. Hypergraphs capture the structured similarities among apps and libraries, matrix factorization learns low-order usage preference, while the propagation module models multi-hop dependencies. These are integrated in a data-driven fusion

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT APPROACHES

Dataset	Approaches	K = 5					K = 10				
		MP	**MR**	**MF**	**MAP**	**COV**	**MP**	**MR**	**MF**	**MAP**	**COV**
$r_n = 1$	POP [10]	0.0753	0.3765	0.1255	0.2840	0.0316	0.0457	0.4565	0.0831	0.2949	0.0465
	LibRec [1]	0.1267	0.6335	0.2112	0.4622	0.2921	0.0668	0.6682	0.1215	0.4669	0.2990
	CrossRec [2]	0.0031	0.0155	0.0052	0.0061	0.0472	0.0069	0.0687	0.0125	0.0130	0.0783
	LibSeek [3]	0.1348	0.6741	0.2247	0.5236	0.3346	0.0755	0.7553	0.1373	0.5346	0.3960
	GRec [6]	0.1521	0.7607	0.2536	0.6269	0.6948	0.0828	0.8283	0.1506	0.6360	0.7918
	AMF-GR	**0.1585**	**0.8012**	**0.2651**	**0.6684**	**0.8726**	**0.0851**	**0.8527**	**0.1549**	**0.6741**	**0.9254**
$r_n = 3$	POP [10]	0.2147	0.3579	0.2684	0.5931	0.0322	0.1341	0.4468	0.2063	0.5682	0.0455
	LibRec [1]	0.2789	0.4648	0.3486	0.6883	0.2916	0.1542	0.5142	0.2373	0.6864	0.2936
	CrossRec [2]	0.0187	0.0312	0.0234	0.0299	0.0896	0.0220	0.0734	0.0339	0.0439	0.1508
	LibSeek [3]	0.3710	0.6183	0.4637	0.7280	0.3245	0.2158	0.7193	0.3320	0.6971	0.3907
	GRec [6]	0.4099	0.6915	0.5142	0.7977	0.6849	0.2337	0.7879	0.3602	0.7605	0.7824
	AMF-GR	**0.4388**	**0.7511**	**0.5539**	**0.8291**	**0.8623**	**0.2429**	**0.8224**	**0.3741**	**0.7986**	**0.9118**
$r_n = 5$	POP [10]	0.3383	0.3383	0.3383	0.7413	0.0316	0.2180	0.4360	0.2907	0.6813	0.0449
	LibRec [1]	0.4400	0.4400	0.4400	0.6922	0.2885	0.2434	0.4868	0.3245	0.6890	0.2992
	CrossRec [2]	0.0342	0.0342	0.0342	0.0596	0.1701	0.0371	0.0743	0.0495	0.0780	0.2560
	LibSeek [3]	0.5291	0.5291	0.5291	0.7896	0.3141	0.3293	0.6587	0.4391	0.7396	0.3796
	GRec [6]	0.5868	0.5945	0.5902	0.8297	0.6571	0.3613	0.7312	0.4834	0.7856	0.7536
	AMF-GR	**0.6314**	**0.6422**	**0.6355**	**0.8599**	**0.8018**	**0.3785**	**0.7624**	**0.5052**	**0.8127**	**0.9021**

TABLE II
WILCOXON SIGNED-RANK TEST RESULTS WITH BONFERRONI CORRECTION

Comparison	P-value	Effect Size	Alpha (Corrected)	Sig. Diff.
AMF-GR vs POP [10]	0.000488	0.8832	0.01	Yes
AMF-GR vs LibRec [1]	0.000488	0.8832	0.01	Yes
AMF-GR vs CrossRec [2]	0.000488	0.8832	0.01	Yes
AMF-GR vs LibSeek [3]	0.000488	0.8832	0.01	Yes
AMF-GR vs GRec [6]	0.000488	0.8832	0.01	Yes

TABLE III
FRIEDMAN TEST RESULTS

Metric	H-stat	P-value	Alpha
MP	24.00	0.00022	0.05
MR	24.00	0.00022	0.05
MF	24.00	0.00022	0.05
MAP	24.00	0.00022	0.05
COV	24.00	0.00022	0.05

step toward a joint representation space, which brings together preference tendencies and structural dependencies. This forms the basis of the recommendation results discussed in the next section.

IV. EXPERIMENTAL EVALUATION

We evaluate the effectiveness of our approach by comparing it with five state-of-the-art third-party library recommendation methods. We selected these baselines because they represent a wide range of strategies, including popularity-based heuristics, collaborative filtering, cross-domain reasoning, matrix factorization, and graph neural networks. By trying AMF-GR on this diverse set, we can better see how accurate, reliable, and varied its recommendations are.

- **POP** [10]: Always recommends the libraries most frequently used, irrespective of the characteristics of the app.
- **LibRec** [1]: A hybrid method that integrates collaborative filtering with association rule mining to capture co-usage patterns.
- **CrossRec** [2]: A collaborative filtering approach between domains aimed at transferring similarity patterns between projects.
- **LibSeek** [3]: A matrix-factorization-based technique specifically designed for Android library recommendation.
- **GRec** [6]: A graph neural network model that learns higher-order relationships through pairwise edges on an app–library graph.

To ensure a fair comparison, the parameter settings of all baselines follow their original configurations reported in [1]–[3], [6]. AMF-GR uses 128 latent factors and three hypergraph propagation layers, selected through preliminary validation. Table I summarizes the average performance in Mean Precision (MP), Mean Recall (MR), Mean F1 (MF), Mean Average Precision (MAP) and Coverage (COV).

Across all values of K and repetition settings r_n, AMF-GR consistently achieves the highest scores. The improvements are substantial: for example, at $K = 5$ and $r_n = 1$, AMF-GR improves MP by 4.2% and MAP by 6.6% over GRec. Similar gains appear for $K = 10$, where AMF-GR maintains a 3-5% advantage in MP and MAP and achieves the highest coverage values in all configurations. Consistent superiority over LibRec and CrossRec, yielding 20% to 50% higher MP and MAP, shows the limitations of methods relying solely on co-usage frequencies or similarity heuristics. AMF-GR outperforms POP by more than 100% in numerous scenarios,

and it greatly minimizes the influence of popularity bias by suggesting a wider and more varied set of libraries. Combining low-order multivariate feature signals with high-order hypergraph propagation enables the Adaptive Multivariate Fusion Graph Reasoning model to capture a more complex relational structure than existing methods.

To formalize the evaluation, we test the following hypotheses.

- *H0: There is no significant difference between AMF-GR and the other methods.*
- *H1: AMF-GR performs significantly better than the other methods.*

To verify that the observed improvements are statistically significant, First, we performed the Friedman test [11] on all models and metrics to ensure statistically significant improvements. The results show that the null hypothesis ($p < 0.001$) is convincingly rejected, and indicating that there are significant differences among the six methods. We then performed a series of pairwise Wilcoxon signed-rank tests with Bonferroni correction ($\alpha = 0.01$) to compare AMF-GR directly with each baseline. The results in Table II show that all comparisons yield $p < 0.05$ with large effect sizes, which implies that the improvements given by AMF-GR are both statistically significant and practically relevant. Overall, these findings confirm that AMF-GR provides better recommendation accuracy, ranking performance, and higher coverage than previous methods consistently.

A. Ablation Study and Complexity Analysis

To explore the contribution of each component, we conduct an ablation study by removing the key modules from AMF-GR. We compare four variants in our experiments: the *Full Model*, which includes the matrix factorization encoder, hypergraph propagation, and adaptive fusion; a variant without the *hypergraph encoder*, reducing the model to pure matrix factorization; a variant without the *matrix factorization* branch, relying solely on hypergraph propagation; and a variant using *fixed weight fusion* ($\alpha = 0.01$), which disables the model's ability to learn the balance between lowand high-order signals.

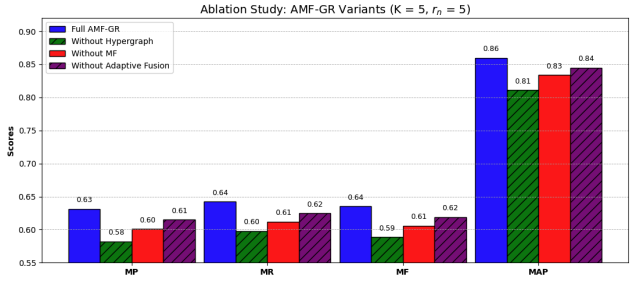

Fig. 2. Ablation Study: Performance Impact of Removing Key Components from AMF-GR at $r_n = 5$, $K = 5$.

Figure 2 illustrates that the Full Model achieves the best performance of all metrics. Among these ablations, removing the hypergraph encoder incurs the most significant loss, especially on MF and MAP, which directly verifies that the high-order structural information is essential. Removing the MF branch also decreases the performance, which shows that the direct co-usage signal is informative. Utilizing a fixed fusion weight results in moderate but consistent degradation, confirming that the optimal relative contributions of MF and hypergraph representations vary across applications and are best learned adaptively.

TABLE IV
COMPUTATIONAL COMPLEXITY COMPARISON

Model	Training Time (s)	Memory Usage (MB)
AMF-GR	1,031.8	616
GRec [6]	13,456	778
LibSeek [3]	314.1	12,670

Besides accuracy, we further analyze the computation cost of AMF-GR for a practical evaluation on its efficiency. We report training time and GPU memory usage for AMF-GR and key baselines in Table IV. AMF-GR finishes training in 1,031.8 seconds using 616 MB of GPU memory, which is much faster than GRec [6] and far more memory-efficient than LibSeek [3]. This shows that AMF-GR delivers its benefits without adding much computational overhead.

B. Threats to Validity

When we assess our approach, we consider both internal and external threats to validity. These factors show how much we can trust our findings and how likely they are to hold up in other settings.

Internal validity Reported performance can shift based on how we configure the model and design our experiments. Some hyperparameter choices can introduce bias, and different settings can change how the model converges or how well it ranks results. Practical details like hardware, operating system, or libraries can also affect learning. To reduce these risks, we run the full experimental workflow in Docker so the software environment stays consistent. Relying on a single dataset is another issue, since MALib's app mix, dependency patterns, and sparsity may favor certain structural traits.

External validity Different software ecosystems, such as iOS, npm, and PyPI, show unique dependency patterns, varying levels of sparsity, and their own evolutionary traits. These variations may influence how recommendation models behave when applied outside the domain in which they were originally evaluated. Real software systems also evolve over the time as libraries become deprecated, APIs change, and dependency graphs shift, which can affect the stability of models trained on historical data. Evaluating AMF-GR on additional ecosystems and under varying temporal conditions would help strengthen its robustness to domain shift and evolving dependency structures. In light of these threats, our aim is to offer a transparent evaluation of the limitations in this work and to assist future efforts to validate the AMF-GR in more diverse environments.

V. Discussion

The results demonstrate that the proposed method indeed provides clear gains over the state-of-the-art recommendation techniques due to jointly modeling both direct co-usage patterns and high-order structural relationships by mixing matrix factorization with hypergraph propagation. Seeing everything at once helps the model notice how apps work in similar ways and pick libraries that fit the bigger picture. This leads to higher accuracy, better rankings, and wider coverage. The adaptive fusion step learns how much to rely on MF or hypergraph signals for each app, which reduces popularity bias and supports more diverse, context-aware recommendations. The ablation study further confirms that the contributions of the hypergraph encoder and the fusion gate are the most substantial towards the final performance.

Despite its effectiveness, The proposed approach currently operates at the library level without considering the version-specific information or the temporal evolution in dependency usage. Both aspects matter for real-world deployment, especially in fast-changing Android environments. Adding version-aware reasoning or time-sensitive interaction patterns is a promising way to make the framework more useful in practice. Even with these limits, AMF-GR still offers a strong base for next-generation TPL recommendation by bringing together simple preferences and complex relational signals in one clear, interpretable design.

VI. Conclusion and Future Work

This work presents AMF-GR, an Adaptive Matrix Factorization and Graph Fusion framework for recommending third-party libraries in Android apps. It blends simple user actions with richer structural data to reveal connections that typical collaborative filtering and graph methods often overlook. Tests on a large real-world dataset show consistent improvements in precision, ranking, and the variety of recommendations. An ablation study further reveals that each component of the framework is essential to its overall performance.

In future work, we plan to extend AMF-GR in several ways. Adding information about library versions and how dependencies evolve over time can make the recommendations more accurate and tailored to each project. Testing our approach in other ecosystems, such as iOS, Python, and Java, will help us see how well it generalizes beyond our current environment. Working closely with developers, gathering feedback, and updating the system step by step can also make it more helpful in everyday practice. Altogether, these directions set up AMF-GR as a strong foundation for continued research on automated library recommendations in modern software engineering.

Acknowledgment

The authors acknowledge the funding received under the SPARC and ISEA schemes of the Government of India.

References

[1] F. Thung, D. Lo, and J. Lawall, "Automated library recommendation," in *2013 20th Working conference on reverse engineering (WCRE)*. IEEE, 2013, pp. 182–191.

[2] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, and M. Di Penta, "Crossrec: Supporting software developers by recommending third-party libraries," *Journal of Systems and Software*, vol. 161, p. 110460, 2020.

[3] Q. He, B. Li, F. Chen, J. Grundy, X. Xia, and Y. Yang, "Diversified third-party library prediction for mobile app development," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 150–165, 2020.

[4] B. Li, H. Quan, J. Wang, P. Liu, H. Cai, Y. Miao, Y. Yang, and L. Li, "Neural library recommendation by embedding project-library knowledge graph," *IEEE Transactions on Software Engineering*, 2024.

[5] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[6] B. Li, Q. He, F. Chen, X. Xia, L. Li, J. Grundy, and Y. Yang, "Embedding app-library graph for neural third party library recommendation," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 466–477.

[7] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," *IEEE transactions on knowledge and data engineering*, vol. 35, no. 6, pp. 5879–5900, 2022.

[8] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang, "Hypergraph contrastive collaborative filtering," in *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 70–79.

[9] A. Jamwal and S. Kumar, "Third-party library recommendations through robust similarity measures," in *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE*, INSTICC. SciTePress, 2025, pp. 635–642.

[10] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–45, 2014.

[11] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013.

[12] C. Teyton, J.-R. Falleri, and X. Blanc, "Automatic discovery of function mappings between similar libraries," in *2013 20th working conference on reverse engineering (WCRE)*. IEEE, 2013, pp. 192–201.

[13] A. Jamwal and S. Kumar, "Towards an approach for project-library recommendation based on graph normalization," in *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE*, INSTICC. SciTePress, 2025, pp. 120–127.

[14] B. Rosner, R. J. Glynn, and M.-L. T. Lee, "The wilcoxon signed rank test for paired comparisons of clustered data," *Biometrics*, vol. 62, no. 1, pp. 185–192, 2006.

[15] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, L. Ochoa, T. Degueule, and M. Di Penta, "Focus: A recommender system for mining api function calls and usage patterns," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1050–1060.

[16] L. He, L. Bai, X. Yang, H. Du, and J. Liang, "High-order graph attention network," *Information Sciences*, vol. 630, pp. 222–234, 2023.

[17] Y. Wang, Q. He, D. Ye, and Y. Yang, "Formulating criticality-based cost-effective fault tolerance strategies for multi-tenant service-based systems," *IEEE Transactions on Software Engineering*, vol. 44, no. 3, pp. 291–307, 2017.

[18] Z. Liu, X. Xia, D. Lo, and J. Grundy, "Automatic, highly accurate app permission recommendation," *Automated Software Engineering*, vol. 26, pp. 241–274, 2019.

[19] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4503–4511.