Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 28 (2025), No. 3, pp. 931–944
DOI : 10.47974/JDMSC-2204

Graph-based recommendation algorithm for personalized suggestions

Rashmi Singh [§]
Department of Mathematics
Amity Institute of Applied Sciences
Amity University
Noida 201301
Uttar Pradesh
India

Khursheed Alam [†]
Neha Bhardwaj *
Department of Mathematics
Sharda School of Basic Sciences & Research
Sharda University
Noida 201310
Uttar Pradesh
India

Abstract

In the present study, we put forward a graph-based recommendation algorithm designed to provide personalized suggestions to users. By leveraging principles from graph theory, our algorithm utilizes the inherent structure for the recommendation system. We consider user preferences and item similarities to construct a graph representation of the recommendation network. Through a combination of graph traversal and similarity analysis techniques, our algorithm identifies relevant items for each user based on their preferences and the connections between items. The personalized suggestions generated by our algorithm aim to improve the overall user experience by offering tailored recommendations that align with individual interests.We gave a case study and for the understanding of our algorithm that may also help in comparing it against existing recommendation methods. This research contributes to the field of recommendation systems by employing graph theory principles to enhance the accuracy and personalization of the recommendation process. Also, we discuss

[§] *E-mail:* rsingh7@amity.edu
[†] *E-mail:* khursheed.alam@sharda.ac.in
* *E-mail:* nehabhr1807@gmail.com (Corresponding Author)

graph homomorphism and graph isomorphism protocols to create encryption methods in cryptography.

Subject Classification (2020): *94C15, 68P25.*

Keywords: *Graph based recommendation, Graph theory, Graph traversal.*

1. Introduction

Graph theory has proven to be a valuable tool in understanding and addressing various aspects of cybersecurity and related fields. Khaleel and Al-Shumam [1] conducted a comprehensive study on graph theory applications in information technology security, highlighting its relevance in analyzing network structures, identifying vulnerabilities, and optimizing security measures. Kuk, Milić, Spalević, and Gocić [2] delved into algorithm design in Python specifically for cybersecurity, emphasizing the importance of efficient and effective algorithms in combating cyber threats.

Sheth, Bhosale, and Kurupkar [3] contributed to the field through their research on contemporary cybersecurity topics in India, shedding light on the latest developments and challenges faced in the realm of cybersecurity. Li, Voos, Darouach, and Hua [4] delved into the practical implementation of linear algebra theory within the realm of networked control systems. Saini [5] emphasized the importance of mathematical modeling and simulation in cyber defense, highlighting their potential in analyzing attack patterns, predicting vulnerabilities, and developing robust security measures. Pandey [6] addressed the issues and challenges associated with modern network security, underlining the need for advanced techniques to safeguard critical infrastructure and systems. In [7], Kumar, and Sharma have proposed research that includes the concepts of cellular automata to increase strength by suggesting proposed keys to encode and decode the information and analyzing image pixels intensity.

Hassan and Chickade [8] reviewed graph coloring techniques as a means to mitigate interference in wireless networks and improve network performance. In [9], authors focused on cryptanalysis and improvements on graph-based authentication schemes, contributing to the field of secure authentication methods. Sen and Samanta [10] discussed network security using graph theory, emphasizing its potential in modeling network structures, detecting anomalies, and enhancing system resilience. In [11], Ghadbane introduced a cryptosystem based on the word problem in a

group *G*. Additionally, Singh and Umrao [12] investigated the concept of finite order nearness in soft set theory, providing a mathematical foundation for analyzing security-related factors.

Graph theory has been widely applied in diverse domains, including process design and analysis. Mah [13] explored the application of graph theory to process design and analysis, highlighting its potential in optimizing complex systems. Thulasiraman and Swamy [14] provided an extensive overview of graph theory and algorithms, offering foundational knowledge for the development of graph-based algorithms. Authors proposed algorithm in [15] using the modified logistic map for the generation of chaotic sequence, encrypts the image in single scan, that makes it to run faster and making the proper use of confusion and diffusion.

While graph theory has found broad application, it is important to critically assess its relevance in specific contexts. Kannaiyan, Pappula, and Veerubommu [16] provided insights into its strengths and limitations and hence providing a review on graph theory. This research paper builds upon these existing studies, focusing specifically on the development of a graph-based recommendation algorithm. Berge's seminal works, "The Theory of Graphs and its Applications" [17], serve as foundational references for graph theory.

Various papers based on graph theory applications have been studied and the usage of Graph theory is explored in cryptography [18]. In [19], Perera and Wijesiri proposed a new connection between graph theory and symmetric cryptography to protect the information from the unauthorized parties. This proposed methodology used a matrix as the secret key which adds more security to the cryptosystem. Graph theory concepts may be applied in various fields of mathematics. A fuzzy set with an interval is an extension of the idea of a fuzzy set. In comparison to fuzzy models, interval-rated fuzzy models provide the system better precision, flexibility and compatibility. Pius and Kirubaharan [20] presented the idea of encryption to secure fluffy computations in fluffy graph.

This paper proposes and gives a method to evaluates a graph-based personalised recommendation algorithm. The graph theory-based method improves recommendation system accuracy and relevance. The algorithm graphs the recommendation network using user preferences and item similarity. Based on user choices and item connections, the system finds relevant things via graph traversal and similarity analysis. The algorithm's personalised suggestions improve user experience by matching interests. As a future scope we propose fuzzy graph theory that applies fuzzy sets

to graph theory. Fuzzy graph theory allows for a more flexible representation of uncertainty in graph-related problems, such as graph colouring or shortest path algorithms. Fuzzy Sets have been used to generalize various concepts, and generalization in nearness spaces [12, 21–23].

This article is structured as follows: In Section 2, we examine the pertinent mathematical principles and procedures of graph theory. Those specifically involved with the procedure. The readers' familiarity with Python programming is presumed, as it is necessary for the implementation of the proposed model discussed in this article. The focus of Section 3 is the presentation of our proposed model, which is implemented with Python. Section 4 discusses the case study, we provide a comprehensive explanation of the algorithm's operation and demonstrate its efficacy. In Section 5, graph-based cryptographic protocols used for securing encryption schemes are discussed. The findings of the study are summed up in Section 6, and prospective directions for future research are outlined in the concluding section of the report.

2. Basic concepts

2.1 A graph can be defined mathematically as a tuple $G = (V, E)$, where: V represents the set of vertices or nodes in the graph. It can be denoted as $V = \{v_1, v_2, v_3, ..., v_n\}$, where vi represents an individual vertex. E represents the set of edges or connections between the vertices. It can be denoted as $E = \{e_1, e_2, e_3, ..., e_m\}$, where ei represents an individual edge. Each edge e_i is associated with a pair of vertices (v_i, v_j), indicating that there is a connection or relationship between vertex v_i and vertex v_j.

2.2 A weighted graph extends this definition by including the set of weights, denoted as W, associated with each edge in the graph. It can be denoted as $G = (V, E, W)$, where $W = \{w(e_1), w(e_2), w(e_3), ..., w(e_m)\}$ represents the set of weights assigned to the edges.

2.3 Graph Traversal refers to the process of systematically exploring or visiting all the vertices/nodes in a graph. It involves moving from one vertex to another while following the edges of the graph. Graph traversal algorithms are used to traverse and search through the graph efficiently. Mathematically, graph traversal can be represented as a function that visits all the vertices in a graph. There are two common ways of graph traversal: breadth-first search (BFS) and depth-first search (DFS).

2.4 Let X and Y be two objects or entities. Similarity analysis can be defined as the process of computing a similarity score $S(X, Y)$ between X and Y using an appropriate similarity measure $sim(X, Y)$ that calculates the similarity between X and Y based on their attributes or features. The similarity score $S(X, Y)$ is the output of this function and represents the degree of similarity between X and Y. Higher scores imply a greater degree of similarity, whilst lower scores suggest a greater degree of dissimilarity. The interpretation of the similarity score depends on the particular similarity measure used.

2.5 *The Precision and Recall is given by*

$$P = TP / (TP + FP),$$

$$R = TP / (TP + FN),$$

Here, TP denotes the count of true positives, which refers to the items correctly identified as relevant by the algorithm. FP represents the count of false positives corresponding to the items incorrectly identified as relevant.

3. Model description

3.1 *Proposed python algorithm:*

We develop a graph-based recommendation algorithm to provide personalized recommendations.

Step 1: Load and preprocess the dataset, organizing it into a suitable graph representation using a Python library like NetworkX.

Step 2: Create a weighted graph.

The weight of an edge in a recommendation graph represents the strength of the relationship or similarity between two items.

Step 3: Implement a graph-based recommendation algorithm that incorporates personalized preferences. The algorithm follows these steps:

- It checks if the user exists in the graph.
- It performs personalized PageRank on the graph, considering the user's history as a personalized preference.

- It sorts the items based on the personalized scores obtained from PageRank.
- It returns the top-k recommendations.

Step 4: Utilize graph traversal algorithms (e.g., BFS, DFS, random walk or personalized PageRank) to explore the graph and identify items likely to be preferred by the user.

Step 5: Implement a scoring mechanism that combines user preferences, song similarities, and possibly other factors (e.g., popularity, novelty) to generate recommendation scores for each item.

Step 6: Provide a set of personalized recommendations based on their preferences and the calculated recommendation scores.

Step 8: Evaluate the algorithm's performance using evaluation metrics-precision and recall.

4. Case Study

Suppose there are 20 songs among 10 users. Organizing it into a suitable graph representation we get the result in Figure 1 for music

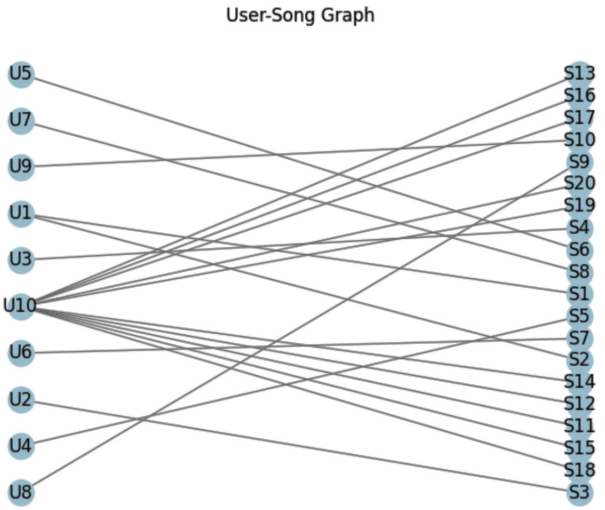

Figure 1

Problem formulated in the form of agraph

recommendations. Here the tuple ('User1', 'Song1') indicates that "User1" is related to "Song1" in the dataset. Similarly, other tuples represent different user-song relationships. The specific criteria for establishing these relationships will depend on the characteristics and structure of your dataset.

Further, we create a weighted graph where nodes represent music items (songs), and edges represent the similarity between songs based on attributes such as genre, artist, or other relevant features. We used different coloured lines for different weights as shown in Figure 2 in the graph visualization.

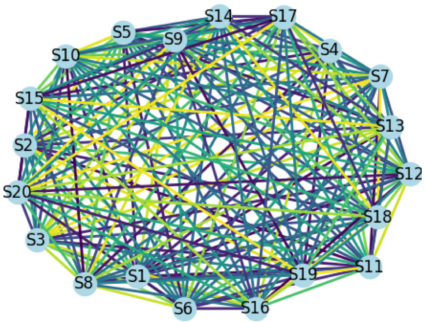

Figure 2

Weighted graph created using figure

On Implementing a graph-based recommendation algorithm that incorporates personalized preferences, we get the output in Figure 3.

```
Recommendations for User 'U1':
S9
S2
S12
S15
S20
S6
S16
S8
S10
S11
S4
S17
S18
S19
S7
S13
S1
S3
S5
S14
```

Figure 3

Personalized music recommendations for one user.

Utilize graph traversal algorithms (we used DFS here, as shown in Fig 4 for user 1, and the same can be done using a random walk or personalized PageRank) to explore the graph and identify music items that are most likely to be chosen.

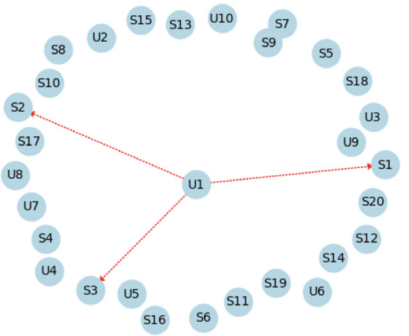

Figure 4

Dotted lines shows the DFS traversal for User 1

Figure 5 shows the scoring obtained in our case that combines user preferences, song similarities, and possibly other factors (e.g., popularity, novelty) to generate recommendation scores for each music item.

```
S1: 24.463499999999996
S2: 28.513999999999996
S3: 31.552999999999994
S4: 33.640999999999984
S5: 34.798
S6: 35.044
S7: 33.95349999999999
S8: 32.257499999999986
S9: 29.102999999999994
S10: 25.603500000000004
S11: 2.484
S12: 2.403
S13: 2.331
S14: 2.2680000000000002
S15: 2.2140000000000004
S16: 2.6235
S17: 2.5875000000000004
S18: 3.033
S19: 2.5335
S20: 2.484
```

Figure 5

Scores for each song

Provide a set of personalized music recommendations for a given user based on their preferences and the calculated recommendation scores, as shown in Figure 6.

```
User U1: ['S7', 'S9', 'S4', 'S6', 'S10', 'S8', 'S1', 'S3']
User U2: ['S1', 'S7', 'S3', 'S10']
User U3: ['S2', 'S8', 'S10', 'S1', 'S7']
User U4: ['S7', 'S2', 'S6', 'S8', 'S10', 'S1']
User U5: ['S1', 'S9', 'S3', 'S5', 'S10', 'S8']
User U6: ['S2', 'S5', 'S8', 'S6', 'S9']
User U7: ['S10', 'S8']
User U8: ['S7', 'S4', 'S9', 'S10', 'S3', 'S6', 'S8']
User U9: ['S6', 'S8', 'S7', 'S4', 'S9', 'S5']
User U10: ['S8', 'S1', 'S6', 'S10', 'S5']
User U11: ['S9', 'S1', 'S7', 'S6', 'S10', 'S8']
User U12: ['S5', 'S8', 'S2', 'S7', 'S4']
User U13: ['S2', 'S4', 'S6']
User U14: ['S7', 'S3', 'S9', 'S2']
User U15: ['S1', 'S4', 'S5', 'S2', 'S8', 'S6']
User U16: ['S4', 'S2', 'S6', 'S3', 'S5']
User U17: ['S7', 'S3', 'S2', 'S4', 'S9']
User U18: ['S9', 'S6', 'S7', 'S1', 'S4']
User U19: ['S10', 'S2', 'S7', 'S1', 'S6', 'S5']
User U20: ['S10', 'S7', 'S8', 'S4', 'S1', 'S3']
```

Figure 6

User Preferences of songs for each user

Evaluate the algorithm's performance using evaluation metrics. To evaluate the algorithm's performance using precision and recall metrics, we need to compare the recommended songs to the relevant songs (ground truth) for a given user the result can be obtained as in Figure 7.

```
Average Precision: 1.0
<function print>
```

Figure 7

Precision value obtained

5. A view on Graph-based Cryptographic Protocols

Graph-based cryptographic protocols use the structural properties of graphs to design secure encryption schemes. By utilizing concepts such as graph homomorphism and graph isomorphism, these protocols aim to create encryption methods that are hard to break due to the computational complexity of graph-related problems.

Step 1: Graph Construction

1. Represent the plaintext data as a graph $G_1 = (V_1, E_1)$, where V_1 represents the vertices (data points) and E_1 represents the edges (relationships between data points).
2. Create a key graph $G_2 = (V_2, E_2)$ with a similar structure to G_1, but using different vertex and edge labels.

Step 2: Graph Homomorphism

3. Define a homomorphic mapping f such that for every edge $(u, v) \in E_1$, there exists a corresponding edge $(f(u), f(v)) \in E_2$. This mapping is the basis for encryption.

Step 3: Encryption

4. Encrypt each vertex $v \in V_1$ to $f(v) \in V_2$.
5. Encrypt each edge $(u, v) \in E_1$ to $(f(u), f(v)) \in E_2$.

Step 4: Decryption

6. Use the inverse homomorphism f^{-1} to decrypt the vertices and edges, reconstructing the original plaintext graph G_1 from the encrypted graph G_2.

Step 5: Security Analysis

7. The security of this scheme relies on the computational difficulty of reversing the graph homomorphism without knowledge of the mapping function *f*.

Case Study: Implementing a Graph-based Encryption Protocol Represented as graph G_1 with vertices as data points (e.g., characters in a message) and edges as relationships (e.g., character adjacency).

Implementation Steps
Graph Construction

- Plaintext graph G_1 for the message "HELLO":
  - Vertices: H, E, L, O
  - Edges: (H, E), (E, L), (L, L), (L, O)
- Key graph G_2 with different labels but a similar structure:
  - Vertices: A, B, C, D
  - Edges: (A, B), (B, C), (C, C), (C, D)

Graph Homomorphism Define a homomorphic mapping f:

- $H \rightarrow A$
- $E \rightarrow B$
- $L \rightarrow C$
- $O \rightarrow D$

Encryption

- Encrypted vertices: A, B, C, D
- Encrypted edges: (A, B), (B, C), (C, C), (C, D)

Decryption

Use the inverse mapping f^{-1} to reconstruct the original message graph G_1.

5.1 *Evaluation*

For the evaluation process, we first verify that the encrypted graph G_2 can be correctly decrypted back to G_1. Analyze the difficulty of determining the mapping f without prior knowledge, relying on the computational hardness of the graph isomorphism problem. The decryption process accurately reconstructs the original message graph, demonstrating the effectiveness of the graph-based encryption scheme. The scheme's security is ensured by the complexity of solving the graph isomorphism problem, making unauthorized decryption computationally infeasible.

6. Concluding remarks

This study introduces a graph-based recommendation algorithm for personalised user suggestions using python codes. The method improves suggestion accuracy and relevance by using graph theory. User preferences and item similarity create the recommendation network graph. Based on user choices and item connections, the system uses graph traversal and similarity analysis to find relevant objects. The algorithm's personalised suggestions improve user experience by matching interests. The paper describes the algorithm's Python implementation and model step-by-step.

The research uses graph theory to improve suggestion accuracy and personalisation. It emphasises graph models, centrality metrics, connection analyses, and other graph theory topics like graph traversal etc. As a

potential avenue for future research, we suggest exploring the application of fuzzy graph theory, which incorporates fuzzy sets into graph theory. By incorporating fuzzy sets, this approach offers a more adaptable representation of uncertainty in various graph-related problems, including graph coloring and shortest path algorithms. Investigating the utilization and potential benefits of fuzzy graph theory could enhance our understanding of uncertain scenarios within graph theory and contribute to the development of more robust algorithms and problem-solving techniques.

Declarations of interest

None.

Disclosure statement

There is no potential conflict of interest declared by the authors.

References

[1] A. Shumam Khaleel, "A Study of Graph Theory Applications in IT Security," *Iraqi Journal of Science,* vol. 61, no. 10, pp. 2705–2714 (2020).

[2] K. Kuk, M. Petar, M. Spalevi, and M. Goci, "Algorithm design in python for cybersecurity," *Electrotechnical and Computer Science Conference,* ERK, Slovenia (2019).

[3] A. Sheth, S. Bhosale, F. Kurupkar, and Asst. Prof., "Research paper on cyber security," *Contemporary Research,* pp. 2231–2137 (2021).

[4] Y. Li, H. Voos, M. Darouach, and C. Hua, "An application of linear algebra theory in networked control systems: stochastic cyber-attacks detection approach," *IMA Journal of Mathematical Control and Information*, vol. 33, no. 4, pp. 1081–1102 (2016).

[5] D. K. Saini, "Cyber Defense: Mathematical Modeling and Simulation," *International Journal of Applied Physics and Mathematics*, vol. 2, no. 5, pp. 1–8 (2012).

[6] S. Pandey, "Modern Network Security: Issues and Challenges," *International Journal of Engineering Science and Technology* (IJEST), vol. 3, no. 5, pp. 1–10 (2011).

[7] A. Kumar and S. Kumar Sharma, "Information cryptography using cellular automata and digital image processing," *Journal of Discrete*

Mathematical Sciences and Cryptography, vol. 25, no. 4, pp. 1105–1111 (2022).

[8] M. A. Hassan and A. Chickade, "A Review of Interference Reduction in Wireless Networks Using Graph Coloring Methods," *Wireless Ad Hoc Networks and Sensor Networks*, pp. 3–13 (2011).

[9] H. O. Abdullah and M. Eftekhari, "Cryptanalysis and Improvements on Some Graph-Based Authentication Schemes," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 16, no. 4-5, pp. 297–306 (2013).

[10] S. Sen and S. Samanta, "Network Security Using Graph Theory," *IJIRT,* vol. 1, no. 4, pp. 223–230 (2014).

[11] N. Ghadbane, "On public key cryptosystem based on the word problem in a group," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 25, no. 6, pp. 1563–1568 (2022).

[12] R. Singh and A. K. Umrao, "On Finite Order Nearness in Soft Set Theory," *WSEAS Transactions on Mathematics*, vol. 18, pp. 118–122 (2019).

[13] R. S. H. Mah, "Application of graph theory to process design and analysis," *Computers & Chemical Engineering*, vol. 7, no. 6, pp. 599–612 (1983).

[14] K. Thulasiraman and M. N. S. Swamy, Graphs: Theory and Algorithms, John Wiley & Sons (2011).

[15] B. Pragathi, B. K. Karunakar Rao, L. Shanmukha Rao, and A. Kumar, "Cryptography algorithms for enhancing security in IoT based MAFLYMPPT system under partial shading conditions," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 7, pp. 1991–2003 (2024).

[16] G. N. Kannaiyan, B. Pappula, and R. Veerubommu, "RETRACTED: A Review on Graph Theory in Network and Artificial Intelligence," *Journal of Physics: Conference Series*, vol. 1831, no. 1, p. 12002 (2021).

[17] C. Berge, The Theory of Graphs and Its Applications, Methuen (1963).

[18] P. Amudha, A. C. Charles Sagayaraj, and A. C. Shantha Sheela, "An application of graph theory in cryptography," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 13, pp. 375–383 (2018).

[19] P. A. S. D. Perera and G. S. Wijesiri, "Encryption and decryption algorithms in symmetric key cryptography using graph theory," *Psychology and Education Journal*, vol. 58, no. 1, pp. 3420–3427 (2021).

[20] A. Pius and D. Kirubaharan, "An effective analysis of cryptography and importance of implementing cryptography in fuzzy graph theory," in Proceedings of International Conference on Communication and Artificial Intelligence: ICCAI 2021, pp. 317–327, Springer (2022).

[21] M. Khare and R. Singh, "Complete α-grills and (l, n)-merotopies," *Fuzzy Sets and Systems*, vol. 159, no. 5, pp. 620–628 (2008).

[22] M. Khare and R. Singh, "L-contiguities and their order structure," *Fuzzy Sets and Systems*, vol. 158, no. 4, pp. 399–408 (2007).

[23] X. Chen, P. Yadav, R. Singh, and S. M. N. Islam, "ES Structure based on soft J-subset," *Mathematics,* vol. 853, pp. 1–11 (2023).

Received June, 2024