2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

An Effective Graph-based Music Recommendation Algorithm for Automatic Playlist Continuation

Toshi-hiro Ito
College of Information Science
University of Tsukuba
Ibaraki, Japan
itou-toshihiro@kde.cs.tsukuba.ac.jp

Hiroaki Shiokawa
Center for Computational Sciences
University of Tsukuba
Ibaraki, Japan
shiokawa@cs.tsukuba.ac.jp

***Abstract*—Automatic playlist continuation (APC) is now essential in music streaming platforms that enable users to discover new music tracks and artists with a seamless interface. To achieve attractive user experiences, it is vital to recommend music tracks that meet the users' interests. However, it is difficult for existing recommendation methods to find effective tracks since the platform includes massive music tracks associated with complex property relationships. In this paper, we propose a novel recommendation algorithm for effective APC. To improve the recommendation accuracy, our algorithm excludes unpromising properties by using a biased graph-based search method. Our extensive experiments on real-world playlists clarify that our algorithm outperforms the state-of-the-art methods in terms of recommendation accuracy.**

***Index Terms*—Music recommendation, Personalized PageRank, Graph search**

I. Introduction

Given a set of music tracks, how can we recommend the most attractive k music tracks (or playlists) for users? This work presents an effective automatic playlist continuation (APC) algorithm for music streaming platforms.

As mobile applications advance, music streaming platforms are growing rapidly worldwide, changing how users consume music. On Spotify [1], one of the leading streaming platforms, the number of monthly active users has increased from 30 million to 500 million in the last decade. Since users now face a significant number of music through the platforms, APC plays an essential role in music consumption [2], [4]. APC is a task to recommend music tracks to users by considering the inference of common characteristics or intentions from given playlists. By recommending a set of music tracks for users, APC enables users to discover new music tracks and artists with a seamless interface.

Although APC is an essential building block in music streaming platforms, it has a critical limitation when handling real-world playlists. Specifically, it is challenging for APC to recommend playlists that meet the common interests in given playlists. This is because a playlist can be constructed with multiple possible intentions [4]. In other words, it is a non-trivial task to infer the common characteristics and intentions from the given playlists, making it difficult for APC to recommend music tracks to users. Hence, effective APC methods are required in the streaming platforms.

ASONAM '23, November 6-9, 2023, Kusadasi, Turkey
© 2023 Association for Computing Machinery.
ACM ISBN 979-8-4007-0409-3/23/11…$15.00
https://doi.org/10.1145/3625007.3627322

A. *Existing Approaches and Challenges*

Various approaches have been proposed to address the above limitations to date [3], [7], [8], [11]. One of the most representative approaches is the *collaborative filtering* [4], [8], which is based on similarities of playlists. By using the similarities, for a query playlist, these methods recommend music tracks from the most similar playlists to the query. Although they have already been applied to several platforms [7] due to their effectiveness, they still have the cold start problem, which means that music tracks are never recommended to users if the tracks do not belong to any playlists.

Alternatively, *content-based methods* [3], [5], [11] has been used to resolve the cold start problem. These methods measure similarities among music tracks based on properties and labels associated with the music tracks, *e.g.* artists, released year, etc. If a query playlist includes a specific music track, the methods recommend music tracks that share the same properties as the track included in the query. By using the properties, these methods effectively avoid the cold start problem. However, their recommendation quality is still limited since a music track and a playlist can have more than one property. Thus, it is still a challenging task to develop an accurate recommendation algorithm for APC.

B. *Our Approaches and Contributions*

We aim to achieve accurate music track recommendations for APC on music streaming platforms. We present a novel recommendation method based on graph-based searches to handle the complex properties associated with music tracks effectively. As described earlier, real-world playlists are expected to have more than one property. However, existing content-based methods do not put strength to the importance of the properties, making it difficult for the methods to distinguish attractive music tracks for users.

Based on the above observation, our algorithm attempts to highlight essential properties for APC by using graph-based similarity search techniques. First, we model the relationships among music tracks, user-created playlists, and their properties as a *heterogeneous tripartite graph*. On this graph, for each given query playlist, our algorithm then extracts a *subgraph of interest (SoI)* to highlight properties to be used in recommendations selectively. Finally, on the SoI, it finds k music tracks recommended for APC by Personalized PageRank (PPR) [6], [9] biased by the query playlist. Consequently, our algorithm leads to the following properties:

- **Accurate:** Our algorithm outperforms the state-of-the-art methods proposed in the last few years regarding recommendation accuracy (Section IV-B). We experimentally confirmed that our algorithm achieves better accuracy than those on Spotify datasets.
- **Effective:** We experimentally demonstrated that the key approaches underlying our algorithm effectively enhance the recommendation accuracy (Section IV-C). Specifically, the biased PPR achieves more accurate recommendations than non-biased ones.
- **Nimble:** Our method requires no pre-computations, unlike existing approaches [5]. Thus, our algorithm enables users to request "on-demand" APC (Algorithm 1).

Our algorithm achieves more accurate APC than the state-of-the-art methods on real-world playlist datasets. For instance, our method outputs better recommendation results under all settings examining over 170 thousand music tracks extracted from Spotify. Although APC is useful in real-life music streaming applications, applying APC on practical playlists is difficult. Our accurate recommendation method will enhance many music streaming applications.

II. Preliminary

Here, we briefly define the problem addressed in this paper. Let m be a music track. We denote a set of music tracks in a music streaming platform as $\mathcal{M} = \{m_1, m_2, \dots, m_{|\mathcal{M}|}\}$. A music track $m_i \in \mathcal{M}$ is annotated with a set of properties, $\ell(m_i) = \{\ \ell_{\textit{Artists}}(m_i),\ \ell_{\textit{Albums}}(m_i),\ \ell_{\textit{Genres}}(m_i),\ \ell_{\textit{Years}}(m_i)\}$, each of which represents artists who composed m_i, albums containing m_i, genres of m_i, and year of release of m_i, respectively. The platform allows users to make playlists that are subsets of $\mathcal{M}$ chosen under the users' intentions. We denote a set of playlists in the platform as $\mathcal{P} = \{p_1, p_2, \dots, p_{|\mathcal{P}|}\}$, where p_i is a playlist made by a user such that $p_i = \{m_{i,1}, m_{i,2}, \dots, m_{i,|p_i|}\} \subseteq \mathcal{M}$.

In this paper, we address the APC problem that recommends music tracks in $\mathcal{M}$ for a given query playlist. Here, we define a query playlist as $p_q = \{m_{q,1}, m_{q,2}, \dots, m_{q,|q|}\} \subseteq \mathcal{M}$. The APC problem is defined as follows:

Problem 1 (APC problem)**.** Given a set of music tracks $\mathcal{M}$, a playlist set $\mathcal{P}$, a query playlist p_q, and $k \in \mathbb{N}$, the APC problem is a task to find k music tracks, $\mathcal{M}_k(p_q)$, from $\mathcal{M}\backslash p_q$ such that $\mathcal{M}_k(p_q)$ maximizes $\sum_{m\in\mathcal{M}_k(p_q)} f(m)$, where $f(m)$ is a function to measure a fitness between m and p_q.

Algorithm 1 Proposed APC Algorithm

```
Input: $\mathcal{M}$, $\mathcal{P}$, $p_q$, and $k$;
Output: $\mathcal{M}_k(p_q)$;
    ▷ (Step 1) Tripartite graph construction:
 1: $V \leftarrow \mathcal{M} \cup \mathcal{P} \cup \pi_{\mathcal{M}}$, $E \leftarrow \emptyset$, $G \leftarrow (V, E)$;
 2: for each $p \in \mathcal{P}$ do
 3:     If $m \in p$ then $E \leftarrow E \cup \{(m, p)\}$;
 4: for each $\ell \in \ell(m)$ do
 5:     $E \leftarrow E \cup \{(m, \ell)\}$;
    ▷ (Step 2) SoI extraction:
 6: $V_q \leftarrow \emptyset, E_q \leftarrow \emptyset, G_q = (V_q, E_q)$;
 7: for each $m \in p_q$ do
 8:     for each $u \in N_G(m)$ do
 9:         $V_q \leftarrow V_q \cup N_G(u)$;
10: for each $(u, v) \in E$ do
11:     If $u, v \in V_q$ then $E_q \leftarrow E_q \cup \{(u, v)\}$;
    ▷ (Step 3) Biased PPR:
12: $\mathbf{v} \leftarrow \alpha\mathbf{R}\mathbf{v} + (1-\alpha)\mathbf{b}$ by Definitions 3 and 4;
13: $\mathcal{M}_k(p_q) \leftarrow \emptyset$;
14: while $|\mathcal{M}_k(p_q)| < k$ do
15:     $u \leftarrow$ arg $\max_{i\in V_q} v_i$;
16:     $v_u \leftarrow 0$;
17:     If $m \in \mathcal{M}\backslash p_q$ then $\mathcal{M}_k(p_q) \leftarrow \mathcal{M}_k(p_q) \cup \{u\}$;
18: return $\mathcal{M}_k(p_q)$;
```

Of course, various definitions are available for f to measure the quality of recommendations. As for f, in this paper, we measured how $V_k(q)$ matches the ground truth provided by a music streaming platform (please see Section IV-A for details).

III. Proposed Method

We present our algorithm for APC on a music streaming platform. Algorithm 1 shows an overview of our algorithm composed of three steps. We introduce (Step 1) Tripartite graph construction, (Step 2) SoI extraction, and (Step 3) Biased PPR in Sections III-A, III-B, and III-C, respectively

A. Tripartite Graph Construction

As shown in Algorithm 1 (lines 1-5), our algorithm first constructs *a heterogeneous tripartite graph* from a set of music tracks $\mathcal{M}$ and a set of playlists $\mathcal{P}$ made by users on the music streaming platform. We here formally define the heterogeneous tripartite graph.

Definition 1 (Heterogeneous tripartite graph)**.** Given a music tracks $\mathcal{M}$ and a playlist set $\mathcal{P}$, a heterogeneous tripartite graph, denoted by G, is defined as $G = (V, E)$ such that $V = \mathcal{M} \cup \mathcal{P} \cup \pi_{\mathcal{M}}$ and $E = E_{\textit{playlist}} \cup E_{\textit{property}}$, where $\pi_{\mathcal{M}} = \bigcup_{m\in\mathcal{M}} \ell(m)$, and $E_{\textit{playlist}}$ and $E_{\textit{property}}$ are given by

$$E_{\textit{playlist}} = \{(m, p) \in \mathcal{M} \times \mathcal{P} \mid \exists m \in \mathcal{M}, \exists p \in \mathcal{P}, m \in p\},$$
$$E_{\textit{property}} = \{(m, \ell) \in \mathcal{M} \times \pi_{\mathcal{M}} \mid \exists m \in \mathcal{M}, \exists \ell \in \pi_{\mathcal{M}}, \ell \in \ell(m)\}.$$

Definition 1 indicates that a tripartite graph G is composed of three types of nodes generated from music tracks, playlists, and properties of the music tracks. In Algorithm 1 (line 1), for each element of $\mathcal{M}$, $\mathcal{P}$, and $\pi_{\mathcal{M}}$, our algorithm assigns a single node $u \in V$. Note that each playlist $p \in \mathcal{P}$ is also regarded as a single node even though p contains multiple music tracks. From Definition 1, our algorithm generates two types of edges, $E_{\textit{playlist}}$ and $E_{\textit{property}}$ shown in Algorithm 1 (lines 2-3) and (lines 4-5), respectively. Specifically, $E_{\textit{playlist}}$ is a set of edges bridging between a music track and a playlist, while $E_{\textit{property}}$ is an edge set from a music track to its property.

B. SoI Extraction

In Algorithm 1 (lines 6-11), our algorithm then extracts *subgraphs of interest* (SoI) for a query playlist p_q to highlight the importance of essential nodes of G in succeeding recommendation processes. Formally, the SoI is defined as follows:

Definition 2 (SoI)**.** For a query playlist $p_q = \{m_{q,1}, m_{q,2}, \ldots, m_{q,|p_q|}\}$ and a tripartite graph $G = (V, E)$, the subgraphs of interest (SoI) of p_q, denoted by $G_q = (V_q, E_q)$, is a subgraph of G such that V_q and E_q are computed as

$$\begin{cases} V_q = \bigcup_{m \in p_q} \bigcup_{u \in N_G(m)} N_G(u), \\ E_q = \{(u, v) \in V_q \times V_q \mid (u, v) \in E\}, \end{cases}$$

where $N_G(u) = \{v \in V \mid (u, v) \in E\} \cup \{u\}$.

G_q is a subgraph of G composed of at most two-hop-away nodes [10] from the query. Similar to the collaborative filtering [3], [8], G_q enumerates all music tracks in $\mathcal{M}$ that share at least one property or playlist with a music track in the query (Algorithm 1 (lines 7-9)). From G_q, our algorithm determines which nodes (property or playlist) should be emphasized their importance for APC in the recommendation processes.

C. Biased PPR

Finally, as shown in Algorithm 1 (lines 12-18), our algorithm recommends k music tracks for the query p_q. Our algorithm selectively emphasizes the importance of nodes in G_q extracted in the previous step to achieve effective recommendations. To this end, for a given SoI of p_q, our method generates the following biased preference vector.

Definition 3 (Biased preference vector)**.** Given a set of music tracks $\mathcal{M}$, a set of playlists $\mathcal{P}$, and a SoI $G_q = (V_q, E_q)$, a biased preference vector $\mathbf{b} = (b_1, b_2, \ldots, b_{|V_q|})^\top$ is defined as

$$b_v = \frac{p(v) \cdot \log |N_{G_q}(v)|}{\sum_{v \in V_q} b_v},$$

where $v \in V_q$ and $p(v)$ is given by

$$p(v) = \begin{cases} 0.8 & (v \in \mathcal{M}), \\ 0.2 & (v \in \pi_{\mathcal{M}}), \\ 0 & (\textit{Otherwise}). \end{cases}$$

Definition 3 indicates that our algorithm highlights the importance of nodes in a SoI G_q from two aspects: (1) a number of neighbor nodes and (2) a node type in G_q. A music track node is regarded as important if it is included in many playlists and shares many properties with other music tracks.

In Algorithm 1 (line 12), our method computes the following biased PPR vector using the biased preference vector.

Definition 4 (Biased PPR vector)**.** Let $\mathbf{R} \in \mathbb{R}^{|V_q| \times |V_q|}$ be a column-normalized adjacency matrix of SoI G_q. Given a biased preference vector $\mathbf{b}$ and $\alpha \in [0, 1]$, the biased PPR vector, denoted by $\mathbf{v} = (v_1, v_2, \ldots, v_{|V_q|})^\top$, is defined as

$$\mathbf{v} = \alpha \mathbf{R} \mathbf{v} + (1 - \alpha)\mathbf{b}.$$

Each element of $\mathbf{v}$ is regarded as a similarity of each node in G_q against the biased preference vector $\mathbf{b}$. That is, nodes can be good candidates to recommend for the query playlist p_q if the nodes have the highest values in the biased PPR vector $\mathbf{v}$.

From the biased PPR vector $\mathbf{v}$, in Algorithm (lines 14-18), our algorithm finally outputs k recommended music tracks for the user-specified query playlist p_q.

IV. Experimental Analysis

Here, we experimentally discuss the effectiveness of our algorithm in terms of recommendation accuracy.

A. Experimental Setting

Methods: We compared our proposed algorithm with the following methods including the state-of-the-art algorithms:

- **HIN-MRS:** The state-of-the-art content-based APC method [11] that explores music tracks similar to a query playlist from music-playlist-music paths by using PPR.
- **Collaborative filtering:** The state-of-the-art collaborative filtering APC method [8] that filters music tracks similar to a query playlist from a music-playlist bipartite graph.
- **Content-based method:** The baseline content-based method [3] that measures a similarity between music tracks by counting up the number of shared properties.

All experiments were conducted on a machine with an Apple M1 CPU 3.20 GHz and 16 GiB RAM. Unless otherwise stated, we set $\alpha = 0.4$ for our proposed algorithm.

Datasets: We collected 10,000 publicly available playlists published by Spotify Million Playlist Dataset Challenge [1]. We split 10,000 playlists into two parts. The first part includes 9,500 out of 10,000 playlists to recommend music tracks for APC. This includes 17,0089 music tracks, 35,892 artists, 81,552 albums, 3,865 genres, and 12 release-year labels. The second part includes the remaining 500 playlists used to evaluate the recommendation quality. We denote the 500 playlists as *evaluation playlists*.

Evaluation criteria: For each evaluation playlist, we randomly selected 10 music tracks as a query playlist, and we regarded the rest tracks as the ground truth of music track recommendations for APC. To evaluate the recommendation quality against the ground truth, we employed two evaluation criteria, *MAP (Mean Average Precision)* and *NDCG (Normalized Discounted Cumulated Gain)*. Both criteria fall between 0 and 1, approaching 1 if the recommendation results close to the ground truth. We varied the number of recommended music tracks, k, shown in Problem 1 from 1 to 50.

[1]https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge

TABLE I: MAP by varying $|p_q|$

Methods	$\|p_q\| = 5$	$\|p_q\| = 10$	$\|p_q\| = 15$	$\|p_q\| = 20$	$\|p_q\| = 25$
Proposed method	0.192	**0.246**	**0.260**	**0.258**	**0.276**
HIN-MRS	**0.196**	0.231	0.238	0.243	0.249
Collaborative filtering	0.152	0.166	0.166	0.164	0.165
Contente-based method	0.099	0.122	0.129	0.136	0.145

TABLE II: NDCG by varying $|p_q|$

Methods	$\|p_q\| = 5$	$\|p_q\| = 10$	$\|p_q\| = 15$	$\|p_q\| = 20$	$\|p_q\| = 25$
Proposed method	0.600	**0.661**	**0.667**	**0.674**	**0.682**
HIN-MRS	**0.616**	0.648	0.648	0.656	0.662
Collaborative filtering	0.549	0.570	0.574	0.576	0.578
Contente-based method	0.483	0.511	0.511	0.526	0.545

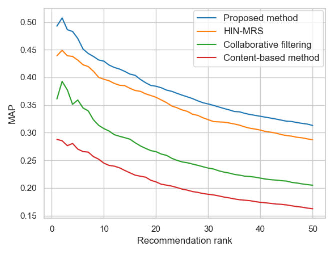


Fig. 1: MAP by varying k

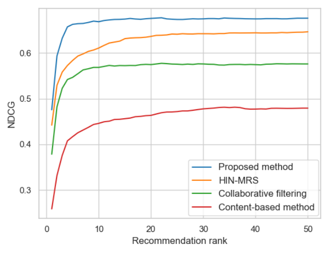
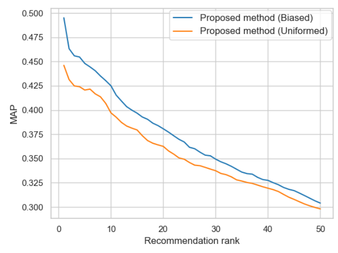
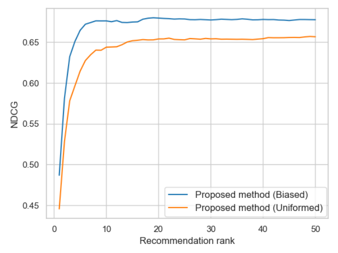


Fig. 2: NDCG by varying k



(a) MAP



(b) NDCG

Fig. 3: Effectiveness of biased preference vector

B. Recommendation Accuracy

We first evaluate the recommendation accuracy. Figures 1 and 2 show MAP and NDCG scores by varying k from 1 to 50. As we can see from the figures, our proposed algorithm significantly outperforms the state-of-the-art methods examined in this paper. As described in Section III, our tripartite graph handles complex relationships among music tracks, properties, and playlists included in a music streaming platform. By contrast, existing methods cannot fully utilize the properties. For example, HIN-MRS uses only music tracks and playlists. Hence, our proposed algorithm effectively improves the recommendation accuracy compared to the other methods.

We then assess the impact of $|p_q|$ against the recommendation accuracy. In Tables I and II, we compared the recommendation accuracy by varying $|p_q|$ as 5, 10, 15, 20, and 25. In this evaluation, we fixed $k = 50$ for each method. The Tables highlight the best results in bold style. As demonstrated in the tables, except for $|p_q| = 5$, our proposed method still keeps the best recommendation accuracy compared to the others. By contrast, HIN-MRS achieves better results than ours if $|p_q| = 5$. Our proposed algorithm enhances the recommendation accuracy using the biased preference vector generated from the query playlist. Hence, if enough large queries are given, our proposed method effectively highlights important properties using the biased preference vector.

C. Effectiveness

In this section, we experimentally discuss the effectiveness of the biased preference vector proposed in Section III-C. We compared MAP and NDCG scores of our proposed algorithm with and without the biased preference vector by varying k from 1 to 50. In Figure 3, we denoted the proposed method using the biased preference vector as "Proposed method (Biased)", while the method without the biased vector is "Proposed method (Uniform)," which uses a uniform vector instead of the biased one. Note that Proposed method (Uniform) is equivalent to PPR used in HIN-MRS.

As we can see from the figures, our biased vector enhances the recommendation accuracy under all the settings examined in this paper. As described in Section I, real-world playlists can have more than one property, making the intention and concept unclear. To overcome this difficulty, our biased vector excludes unpromising properties from graph-based features given by the tripartite graph. Consequently, our proposed algorithm achieves better performances than the others.

V. Conclusion

We proposed a novel music recommendation algorithm for APC on music streaming platforms. Our algorithm constructs a heterogeneous tripartite graph from properties associated with music tracks and selectively emphasizes essential properties to improve recommendation accuracy. Consequently, our proposed method effectively excludes unpromising properties from the recommendations. Our experimental analysis demonstrated that our algorithm outperforms the state-of-the-art methods regarding recommendation accuracy.

Acknowledgment

This work is partly supported by JST PRESTO JPMJPR2033 and JSPS KAKENHI JP22K17894.

References

[1] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas, "Algorithmic Effects on the Diversity of Consumption on Spotify," in *Proc. The Web Conference 2020*, 2020, p. 2155–2165.

[2] G. Bonnin and D. Jannach, "Automated Generation of Music Playlists: Survey and Experiments," *ACM Computing Surveys*, vol. 47, no. 2, 2014.

[3] M. A. Casey, R. C. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-Based Music Information Retrieval: Current Directions and Future Challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[4] A. Ferraro, D. Bogdanov, J. Yoon, K. Kim, and X. Serra, "Automatic Playlist Continuation Using a Hybrid Recommender System Combining Features from Text and Audio," in *Proc. RecSys Challenge 2018*, 2018.

[5] F. Fessahaye, L. Perez, T. Zhan, R. Zhang, C. Fossier, R. Markarian, C. Chiu, J. Zhan, L. Gewali, and P. Oh, "T-RECSYS: A Novel Music Recommendation System Using Deep Learning," in *Proc. ICCE 2019*, 2019, pp. 1–6.

[6] Y. Fujiwara, M. Nakatsuji, T. Yamamuro, H. Shiokawa, and M. Onizuka, "Efficient Personalized PageRank with Accuracy Assurance," in *Proc. KDD 2012*, 2012, pp. 15–23.

[7] Z. Huang, D. Zeng, and H. Chen, "A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce," *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 68–78, 2007.

[8] E. Shakirova, "Collaborative Filtering for Music Recommender System," in *Proc. EIConRus 2017*, 2017, pp. 548–550.

[9] H. Shiokawa, “Fast ObjectRank for Large Knowledge Databases,” in *Proc. ISWC 2021*, 2021, pp. 217–234.

[10] H. Shiokawa, Y. Fujiwara, and M. Onizuka, “SCAN++: Efficient Algorithm for Finding Clusters, Hubs and Outliers on Large-scale Graphs,” *PVLDB*, no. 11, pp. 1178–1189, july 2015.

[11] R. Wang, X. Ma, C. Jiang, Y. Ye, and Y. Zhang, “Heterogeneous Information Network-Based Music Recommendation System in Mobile Networks,” *Computer Communications*, vol. 150, p. 429–437, 2020.