A Case Study of a Transparent and Controllable Music Recommender System with Multi-relational Layers

Kosetsu Tsukuda(✉), Keisuke Ishida, Takumi Takahashi, Masahiro Hamasaki, and Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan
k.tsukuda@aist.go.jp

Abstract. In recommending songs to users, various types of relationships can be considered, such as songs liked by users with similar preferences or songs that are acoustically similar to those the target user already likes. Providing explanations for recommendations based on such relationships improves transparency and trust, but users currently have no control over which relationships are emphasized. To solve this problem, we extend an existing recommendation method based on a graph convolutional network (GCN) by representing each relationship as a separate graph layer with adjustable weights. By applying this method, we implemented a song recommender system with three types of relationships (user preference similarity, acoustic similarity, and creator commonality) on a music web service called "Kiite." On the service, four types of recommendation results are displayed, depending on which relationships are emphasized and to what degree. The recommender system offers both transparency and controllability in that users can freely switch between the four recommendation result types. An analysis of over two years of usage logs demonstrates the effectiveness of combining transparency and controllability in music recommendation.

Keywords: Music recommendation · Graph convolutional network · Web service · User interaction · User behavior analysis

1 Introduction

With the rise of music-streaming and video-sharing services, users now have access to a vast number of songs. Because it is challenging for users to find songs that match their preferences, many music services incorporate recommender systems to estimate and suggest songs that users might like. Such systems often rely on multiple types of relationships, such as "similarity in song preferences between users" and "similarity in acoustic features between songs" [2,13]. Recent research also focuses on improving recommendation accuracy by leveraging multiple relationships [28,36].

Beyond accuracy, transparency is also critical in recommender systems [6,7,29], as it builds user trust [7,21,29] and encourages continued use. A common way to enhance transparency is to provide explanations [4,38]. In music

J. Lokoč et al. (Eds.): MMM 2026, LNCS 16412, pp. 276–291, 2026.
https://doi.org/10.1007/978-981-95-6950-2_20

recommendation, one such approach is to show the relationship used in a recommendation, such as "Users with similar tastes to yours like this song" or "This song is acoustically similar to songs you like."

Although presentation of the reasons for recommendations improves transparency, the relationship used to explain each recommended song is determined by the recommender system. Therefore, if a user sees that most recommended songs come with the explanation "Users with similar tastes to yours like this song," for example, but the user instead wants to see more results based on "This song is acoustically similar to songs you like," that intent cannot be captured in current recommendation systems. The incorporation of controllability, allowing users to choose which relationship to emphasize, could enhance users' interactions and help them discover songs more efficiently, while maintaining transparency.

To address this issue, we propose a recommender system that offers both transparency and controllability. We extend a method based on a graph convolutional network (GCN) [11,35] to handle multiple relationships and emphasize specific ones. Our method introduces a layered structure where each layer represents a different relationship via a graph of vectorized nodes for users, songs, and creators. By assigning weights to each layer and optimizing the node vectors accordingly, the system generates recommendation results that reflect the intended emphasis.

In addition, for a case study, we implemented this method in a music web service called "Kiite"[1] which supports three relationships: "user preference similarity," "acoustic similarity," and "creator commonality." Kiite displays four recommendation result types: three emphasizing a specific relationship, and one combining the relationships equally. Users can freely switch between these types while listening to songs and giving feedback. We analyzed over two years of usage logs from Kiite to examine how users interacted with these recommendation types. Our analysis showed that users who explored more types engaged more actively, and that about 40% of users discovered preferred songs via multiple recommendation types. These findings demonstrate the value of adding controllability to recommender systems.

2 Related Work

In recommender systems, the display of reasons for recommendations to users helps them efficiently find items [23,34], improves system transparency [15,21], and builds trust [5,21,23]. Accordingly, various explanation methods and interfaces have been proposed [4,38]. One approach to explain recommendations is to present estimated user preferences [1,10,34]. For example, Balog et al. [1] generated textual explanations based on tags assigned to movies and presented them alongside recommendations.

[1] https://kiite.jp.

Another approach is to provide explanations for each recommended item [5, 12,15,21,23,32], such as evaluations from similar users [12] or feature similarities with respect to previously liked items [5]. Some methods present multiple explanations based on different relationships [19,20,31]. Kouki et al. [20] used up to seven relationships (e.g., user similarity, metadata similarity, popularity) and offered corresponding explanations. As user preferences for explanation types vary [20,32], the presentation of multiple explanation types can better meet individual needs. However, existing systems typically do not allow users to request more recommendations based on a specific relationship that they prefer.

To address this issue, some studies introduced controllability via slider-based interfaces [3,25,30]. For instance, Bostandjiev et al. [3] allowed users to adjust sliders to control the weights of three relationships. The system then combined results accordingly and presented a unified list. Other studies adopted similar techniques [25,30]. Although sliders offer fine control, users tend to use them more when searching for known items and less when exploring new ones [30].

Our study aims to support new song discovery by presenting recommendations based on multiple relationships. Following prior insights [30], we avoid sliders and instead let users switch between recommendation results that each emphasize a different relationship. This design lowers the user effort and improves the usability relative to slider-based interfaces. When incorporating controllability into a recommender system that considers multiple relationships, prior work has commonly used separate recommendation methods for each relationship [3,25,30]. Alternatively, there have been simple approaches where users choose whether to incorporate meta-information linked to song content into the recommendation results, and filtering is then performed accordingly [14,24,27]. In contrast to those approaches, we leverage the high recommendation accuracy of GCN-based methods [36] and generate recommendations within a unified framework. Moreover, most prior systems offering controllability were evaluated in small, short-term user studies [3,14,24,25,27,30]. In contrast, we implemented our system in a public web service and analyzed logs from 3,264 users over more than two years. This large-scale, long-term analysis highlights how users interact with a recommender system that supports both transparency and controllability.

3 Song Recommendation Method Based on Layered Graph Structure

This section describes our recommendation method that considers various relationships and generates recommendation results with an emphasis on a specific relationship.

3.1 Graph Construction on Layers (Fig. 1(a))

In our method, various relationships considered in recommendation are represented by graphs. A separate graph is constructed for each relationship and treated as a "layer," which enables the generation of recommendation results

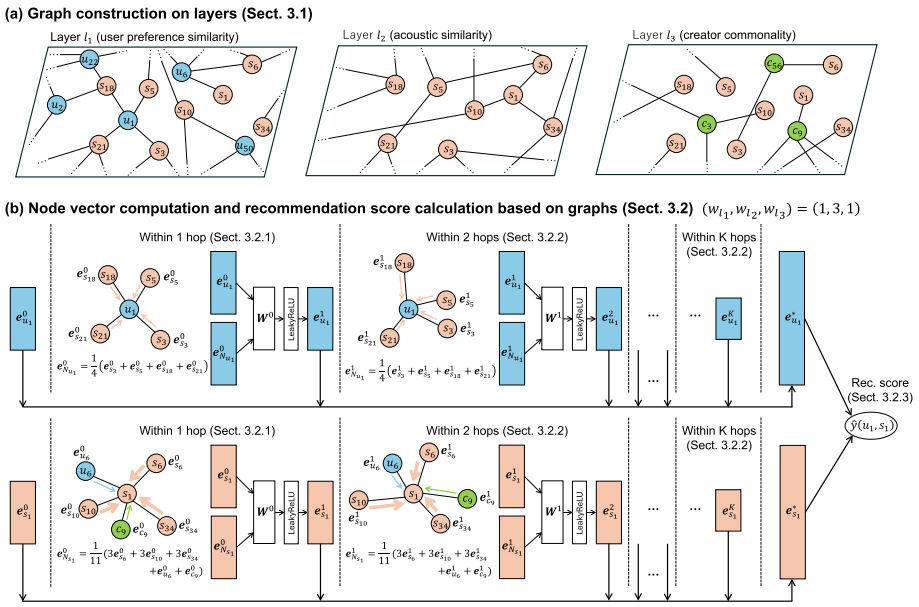

Fig. 1. Method overview.

emphasizing a specific layer. Let L denote the set of layers, and let $G_l = (V_l, E_l)$ denote the graph on a layer $l \in L$, where G_l is an undirected graph, and V_l and E_l represent the sets of nodes and edges in G_l, respectively. For example, for a layer l_1 corresponding to the relationship "user preference similarity," the graph G_{l_1} has nodes $V_{l_1} = U \cup S$ (where U and S are the sets of all users and all songs, respectively), and the edges E_{l_1} are defined between each user $u \in U$ and each song $s \in S$ that the user likes. Here, $\{u, s\}$ denotes an edge between user u and song s.

The set of nodes included in a graph may differ across layers. In addition to the above relationship, suppose we also consider the relationship "acoustic similarity." In this case, the graph on the corresponding layer l_2 is defined as G_{l_2} with $V_{l_2} = S$, and E_{l_2} contains edges between each song $s \in S$ and its similar songs $s' \in S$. Similarly, for the relationship "creator commonality," the graph on the corresponding layer l_3 is defined as G_{l_3} with $V_{l_3} = C \cup S$ (where C is the set of all creators), and E_{l_3} contains edges between each creator $c \in C$ and each song $s \in S$ created by c.[2]

3.2 Graph-Based Vector Calculation for Nodes (Fig. 1(b))

In our method, each node on a graph is represented by a d-dimensional vector $\boldsymbol{e}^0 \in \mathbb{R}^d$, following GCN-based recommendation approaches [36]. A single node

[2] For the rest of this section, we use these three relationships and their corresponding layers (l_1, l_2, l_3) as examples.

that appears in multiple layers is represented by the same vector. For example, a song s that appears in all three layers (l_1, l_2, and l_3) is represented by the same vector $\boldsymbol{e}_s^0$. By using the message-passing architecture [9,37], the value of each node vector is updated according to its neighbors within K-hops in the graph. Below, we first describe how to compute vectors based on 1-hop neighbors; then, we generalize this to k-hops ($1 \leq k \leq K$). Finally, we explain how to compute recommendation scores using the resulting node vectors.

Vector Calculation Based on 1-Hop Neighbors To calculate the vector values of nodes, we assign a weight $w_l \in \mathbb{R}^+$ to each layer $l \in L$. Larger weights are assigned to layers corresponding to relationships that the recommendation should emphasize. Let $N_n^l = \{(t, w_l) \mid \{n, t\} \in E_l\}$ be the set of pairs of neighbor node $t \in V_l$ and weight w_l for node $n \in V_l$ in graph G_l. The union of these sets over all layers is denoted as $N_n = \bigcup_{l \in L} N_n^l$. Then, the vector $\boldsymbol{e}_{N_n}^0$ based on the 1-hop neighbors of node n is computed as follows:

$$\boldsymbol{e}_{N_n}^0 = \sum_{(t,w_l) \in N_n} \pi(n, t, w_l) \boldsymbol{e}_t^0, \quad \text{where} \quad \pi(n, t, w_l) = \frac{w_l}{\sum_{(t', w_l') \in N_n} w_l'}. \tag{1}$$

This equation computes the weighted sum of neighbor vectors, where nodes in layers with higher weights have a greater impact on the value of $\boldsymbol{e}_{N_n}^0$. While existing GCN-based methods use attention mechanisms to automatically learn the importance of each relationship [35], our method enables manual specification of relationship importance to produce recommendation results focused on a specific relationship. Finally, the updated vector $\boldsymbol{e}_n^1$ for node n (according to its 1-hop neighbors, including itself) is computed as:

$$\boldsymbol{e}_n^1 = \text{LeakyReLU}\left(\boldsymbol{W}^0 \left(\boldsymbol{e}_n^0 + \boldsymbol{e}_{N_n}^0\right)\right). \tag{2}$$

In GCNs, this is a common aggregation method for integrating two vectors (in this case, $\boldsymbol{e}_n^0$ and $\boldsymbol{e}_{N_n}^0$) [18]. Here, $\boldsymbol{W}^0 \in \mathbb{R}^{d' \times d}$ is a trainable parameter matrix (with d' typically equal to d or $\frac{d}{2}$), and LeakyReLU [22] introduces nonlinearity.

Vector Calculation Based on k-Hop Neighbors. To compute the vector for node n according to its 2-hop neighbors, we first calculate $\boldsymbol{e}_{N_n}^1$ similarly to $\boldsymbol{e}_{N_n}^0$:

$$\boldsymbol{e}_{N_n}^1 = \sum_{(t,w_l) \in N_n} \pi(n, t, w_l) \boldsymbol{e}_t^1. \tag{3}$$

As $\boldsymbol{e}_t^1$ incorporates the 1-hop neighbors of t, $\boldsymbol{e}_{N_n}^1$ captures the 2-hop neighbors of n. The vector $\boldsymbol{e}_n^2$ for node n is then computed as

$$\boldsymbol{e}_n^2 = \text{LeakyReLU}\left(\boldsymbol{W}^1 \left(\boldsymbol{e}_n^1 + \boldsymbol{e}_{N_n}^1\right)\right). \tag{4}$$

By generalizing this process, we obtain the vector for node n according to its k-hop neighbors:

$$\boldsymbol{e}_n^k = \text{LeakyReLU}\left(\boldsymbol{W}^{k-1} \left(\boldsymbol{e}_n^{k-1} + \boldsymbol{e}_{N_n}^{k-1}\right)\right). \tag{5}$$

Recommendation Score Calculation (Fig. 1(b))

To compute the recommendation score $\hat{y}(u, s)$ for a user u and song s, we first aggregate the vectors for u and s up to K hops across all layers:

$$\boldsymbol{e}_u^* = \boldsymbol{e}_u^0 || \cdots || \boldsymbol{e}_u^K. \tag{6}$$

$$\boldsymbol{e}_s^* = \boldsymbol{e}_s^0 || \cdots || \boldsymbol{e}_s^K. \tag{7}$$

Here, $||$ denotes vector concatenation, which is a commonly used aggregation method in GCNs for incorporating multi-hop information [37].

The recommendation score $\hat{y}(u, s)$ is then computed as the vectors' inner product:

$$\hat{y}(u, s) = \boldsymbol{e}_u^{*\top} \boldsymbol{e}_s^*. \tag{8}$$

3.3 Parameter Learning

For parameter optimization, we adopt the Bayesian Personalized Ranking (BPR) [26]. In BPR, the training data D is defined as follows:

$$D = \{(u, s, s') \mid u \in U \wedge s \in S_u^+ \wedge s' \in S \setminus S_u^+\}, \tag{9}$$

where S_u^+ is the set of songs liked by user u, and the tuple (u, s, s') indicates that u prefers song s over song s'. The loss function based on D is given as

$$L = \sum_{(u,s,s') \in D} - \ln \sigma(\hat{y}(u, s) - \hat{y}(u, s')) + \lambda \|\boldsymbol{\Theta}\|_2^2, \tag{10}$$

where σ denotes the sigmoid function, λ is a hyperparameter, and $\boldsymbol{\Theta} = \{\boldsymbol{E}, \boldsymbol{W}^0, \cdots, \boldsymbol{W}^{K-1}\}$ represents all the model parameters. Note that $\boldsymbol{E}$ is a matrix containinig the initial vectors of all nodes in the layers. For example, in the case of the relationships described in Sect. 3.1, we have:

$$\boldsymbol{E} = \left[\boldsymbol{e}_{u_1}^0, \cdots, \boldsymbol{e}_{u_{|U|}}^0, \boldsymbol{e}_{s_1}^0, \cdots, \boldsymbol{e}_{s_{|S|}}^0, \boldsymbol{e}_{c_1}^0, \cdots, \boldsymbol{e}_{c_{|C|}}^0\right]. \tag{11}$$

4 Application to Music Web Service

By applying our recommendation method, we implemented a recommender system with transparency and controllability on the music web service Kiite. This section first provides an overview of Kiite and then describe how we applied our method to Kiite's data to train the recommendation model. We then explain the recommender system that we implemented using the trained model. While we describe the system with a screenshot of the desktop version of Kiite, there is also a mobile interface allowing users to perform the same operations.

4.1 Overview of Kiite

Song data on Kiite are routinely collected from Nico Nico Douga[3], which is one of the most popular video sharing services in Japan. On Nico Nico Douga, it is quite common for both amateur and professional musicians to upload songs created with singing voice synthesizer software called VOCALOID [16]. As of August 2025, more than 550,000 songs are available on Kiite. Because songs on Nico Nico Douga are published as music videos, songs on Kiite are played using an embedded video player provided by Nico Nico Douga. On Kiite, a registered user can set her own icon image, add songs to her list of favorite songs, create playlists, and listen to other users' playlists.

4.2 Training of Recommendation Model

On Kiite, users can listen to songs through a variety of features. We considered songs that users added to their favorites lists or playlists to be the songs that matched their preferences (i.e., S_u^+ in Sect. 3.3). For the recommendation task, we used the three relationships introduced as examples in Sect. 3.1. In the graph for layer l_1, corresponding to the relationship "user preference similarity," edges were created between a user node $u \in U$ and each song node in S_u^+. In the graph for layer l_2, corresponding to "acoustic similarity," we computed similarity scores using acoustic features [33] obtained as audio embeddings through self-supervised learning, and created edges between each song $s \in S$ and its top 5 most similar songs. Lastly, in the graph for layer l_3, corresponding to "creator commonality," edges were created between each creator node $c \in C$ and each song node $s \in S$ that c created, according to Kiite's metadata.

In addition to a balanced setting of $(w_{l_1}, w_{l_2}, w_{l_3}) = (1, 1, 1)$, we trained three other recommendation models, each emphasizing one relationship: $(w_{l_1}, w_{l_2}, w_{l_3}) = (100, 1, 1)$, $(1, 100, 1)$, and $(1, 1, 100)$. Thus, a total of four recommendation models were trained. All models shared the same training data, hyperparameters (except for the layer weights), and training process. Regarding the hyperparameters, following Wang et al. [35], the dimension d of the node vector $\boldsymbol{e}^0$ was 64, the maximum hop number K was 3, and the parameter matrices W^0, W^1, and W^2 were matrices in $\mathbb{R}^{64\times64}$, $\mathbb{R}^{32\times64}$, and $\mathbb{R}^{16\times32}$, respectively. As a result, both $\boldsymbol{e}_u^*$ in Eq. (6) and $\boldsymbol{e}_s^*$ in Eq. (7) were 176-dimensional vectors $(64+64+32+16)$. We used the mini-batch Adam optimizer [17] for parameter training.

4.3 Recommender System

On Kiite, we use the term "recommendation engine" instead of "recommender system" so that users can more easily understand the concept. In this paper, we use both terms interchangeably. Figure 2 shows the recommendation engine's interface, which incorporates the four recommendation models. For clarity, Kiite

[3] https://www.nicovideo.jp.

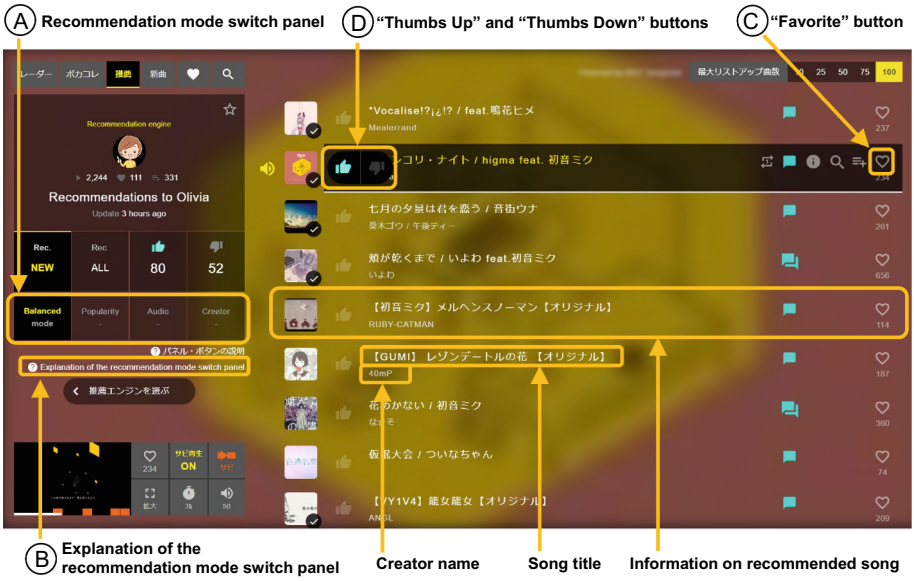

Fig. 2. Interface of the recommendation engine (partially translated into English).

refers to these models as "recommendation modes." To help users distinguish them easily, it uses the following labels:

Balanced Mode: The balanced mode $((w_{l_1}, w_{l_2}, w_{l_3}) = (1, 1, 1))$.

Popularity Mode: The mode emphasizing "user preference similarity" $((w_{l_1}, w_{l_2}, w_{l_3}) = (100, 1, 1))$.

Audio Mode: The mode emphasizing "acoustic similarity" $((w_{l_1}, w_{l_2}, w_{l_3}) = (1, 100, 1))$.

Creator Mode: The mode emphasizing "creator commonality" $((w_{l_1}, w_{l_2}, w_{l_3}) = (1, 1, 100))$.

It is possible to add more layers, such as a layer using tags associated with songs, but previous studies suggest it is preferable to provide around four explanations [20]. Thus, Kiite has only these four recommendation modes.

When a user accesses the recommendation system for the first time, it displays results from the Balanced Mode. The "recommendation mode switch panel" (Fig. 2 Ⓐ) displays toggle buttons for the four modes. When a user selects one, the recommendation results from that mode are shown. Clicking on "Explanation of the recommendation mode switch panel" (Ⓑ) brings up a popup with explanations of each mode, to help users intuitively understand the differences between the modes and to enhance the system's transparency. The popup's descriptions for each mode are as follows:

Balanced Mode: Considers the following three factors in a balanced manner.

Popularity Mode: Emphasizes the popularity of songs on Kiite.

Audio Mode: Emphasizes similarity in the musical characteristics of songs.

Creator Mode: Emphasizes the creators of songs.
Users can compare results across modes or switch to a preferred mode and listen to songs, thus achieving controllability. When users revisit the recommendation engine page, it automatically shows results from the last selected mode, which allows users to easily continue using their preferred mode without switching it manually each time.

The recommendation results include up to 100 songs. Clicking on a song thumbnail or title plays the song. Users can add a song to their favorites list with the "Favorite" button (Ⓒ). They can also provide feedback by clicking either the "Thumbs Up" (relevant) or "Thumbs Down" (irrelevant) button (Ⓓ), with the feedback used later for model updates. Users can listen to entire songs or opt for a more efficient experience by automatically playing only choruses, via chorus detection technology [8].

Kiite updates all four recommendation models once a day according to user interactions. Among the songs with such interactions, those added to favorites lists or marked "Thumbs Up," are treated as positive examples, while those marked "Thumbs Down" are treated as negative examples. Each positive song is added to S_u^+ for user u, and new edges are created between u and those songs in the graph for layer l_1. These newly added edges are then incorporated in vector calculations (e.g., Eq. (1)). Similarly, negative examples are used in the BPR loss calculation (Eq. (10)). Typically, in BPR, songs s' are randomly sampled from $S \setminus S_u^+$, but Kiite samples some s' from the user's negative examples with a certain probability. This helps the model learn parameters more effectively.

When updating the model, the system does not retrain the parameters from scratch; instead, the previous day's parameters are updated incrementally. If there are newly registered users, newly created songs, or first-time creators, their vectors must also be initialized. To efficiently determine the vector values for such new data, rather than initializing them randomly, we use the following strategy: for a new song s, the initial vector is set to the average vector of the top 10 songs most similar to s in terms of acoustic features; for a new user u, the initial vector is set to the average vector of the songs included in S_u^+; and for a new creator c, the initial vector is set to the average vector of the songs created by c.

5 Analysis

For a case study, we released the recommender system described in Sect. 4 on Kiite on November 22, 2022. In this section, we analyze the resulting user interactions with the system, which offers both transparency and controllability. For this purpose, we used over two years of user activity logs collected between November 22, 2022, and December 24, 2024[4]. We focused on 3,264 individual users (i.e., 3,264 recommendation engines) who played at least one song in the recommender system during the analysis period.

Table 1. Number of users who used m different recommendation modes.

$m = 1$	$m = 2$	$m = 3$	$m = 4$
1,503 (46.05%)	533 (16.33%)	404 (12.38%)	824 (25.24%)

5.1 Switching Between Recommendation Modes

Table 1 gives the distribution of the number of recommendation modes used. First, 46.05% users used a single mode ($m = 1$), which indicates that those users never switched modes and only used the default Balanced Mode. As the system automatically creates recommendation engines for all users, it is likely that those users were not very interested in receiving recommendations, stopped using the recommendation engine after briefly exploring it, or simply did not notice they could change the recommendation mode. Nevertheless, among the 3,264 users, 1,761 (53.95%) switched at least once from the initial Balanced Mode. In addition, among those who used two or more modes, the number of users who accessed all four recommendation modes ($m = 4$) was the highest. This was likely because once users started using the recommendation modes, they recognized the usefulness and became interested in exploring and comparing all four.

Next, we analyzed how the number of recommendation modes used (m) affected the frequency of interactions with the recommended songs. The analyzed interaction types were: "Song Plays," "Added to Favorites," "Thumbs Up", and "Thumbs Down." Table 2 lists the results. For example, the cell under "Song Plays" and $m = 2$ shows "51.96 (144.23)," meaning that among the 533 users who used two modes, the average number of song plays was 51.96, with a standard deviation of 144.23.

The results show that, for all interaction types, the average frequency of interaction increased with the number of recommendation modes used. This suggests that users who used more recommendation modes tended to engage more actively

[4] According to Kiite's terms of service, user activity logs may be used for academic research purposes.

Table 2. Average frequency (standard deviation in parentheses) of user interactions by the number of recommendation modes used (m).

Interaction Type	$m = 1$	$m = 2$	$m = 3$	$m = 4$
Song Plays	29.78 (90.55)	51.96 (144.23)	96.64 (262.5)	263.55 (1010.40)
Added to Favorites	3.46 (13.43)	10.26 (39.95)	13.85 (53.40)	37.00 (145.03)
Thumbs Up	7.75 (32.15)	13.07 (37.53)	21.04 (72.42)	62.47 (244.01)
Thumbs Down	7.14 (40.90)	9.87 (52.33)	27.95 (221.34)	91.41 (661.26)

with the system and discovered more songs matching their preferences (i.e., songs added to their favorites). These findings suggest that Kiite could encourage users who only use one mode to try other modes, possibly by displaying a message on the interface. This is an interesting topic for further research: if such messages successfully increased interaction, it would provide evidence of a causal link between controllability in recommendation explanations and increased user engagement.

5.2 Interaction Within Recommendation Modes

In this section, we analyze the distribution of interactions within each recommendation mode to assess whether offering multiple switchable modes adds value. To focus on engaged users, we limited our analysis to 1,228 users who had at least 10 total interactions across three types: "Added to Favorites," "Thumbs Up," and "Thumbs Down."

We first counted the number of users who performed each interaction type at least once per recommendation mode, as well as the total frequency of those interactions. The results are summarized in Table 3. For all interaction types, the numbers of users followed this order: Balanced > Audio > Popularity > Creator.

Next, we analyzed which recommendation mode was most frequently used per interaction type for each user. For instance, if a user played 22 songs in Balanced Mode, 39 in Popularity Mode, 24 in Audio Mode, and 16 in Creator Mode, the most used mode for "Song Plays" would be Popularity Mode for this user. To ensure that the modes were used meaningfully, we only included users who accessed their most-used mode at least three times during a given interaction. The results are given in Table 4. We can see that Balanced Mode was the most preferred across all interaction types, followed by Audio Mode, Popularity Mode, and Creator Mode. Although Popularity and Creator Modes were used less frequently, there were still users who preferred them, thus validating the importance of offering multiple selectable modes.

Finally, to assess whether users actually used multiple modes, we identified the number of modes used at least three times for each interaction. For example, if a user played one song in Balanced Mode, two in Popularity Mode, nine in Audio Mode, and five in Creator Mode, we would regard two modes (i.e., Audio

Table 3. Number of users with at least one interaction per mode (total interactions in parentheses).

Interaction Type	Balanced	Popularity	Audio	Creator
Song Plays	1,180 (138,891)	601 (53,019)	750 (90,888)	442 (21,084)
Added to Favorites	935 (20,739)	389 (7,814)	548 (13,621)	233 (2,765)
Thumbs Up	970 (39,477)	430 (15,704)	612 (18,067)	293 (4,197)
Thumbs Down	638 (41,560)	244 (17,885)	377 (35,813)	174 (7,165)

Table 4. Number of users who used each recommendation mode the most for each interaction type.

Interaction Type	Balanced	Popularity	Audio	Creator
Song Plays	799	127	300	37
Added to Favorites	621	125	217	24
Thumbs Up	698	118	265	31
Thumbs Down	398	55	151	20

and Creator) as being actively used for "Song Plays" by this user. The results are listed in Table 5. For every type of interaction, almost 40% or more of users who performed the interaction at least three times in any recommendation mode actively used two or more recommendation modes. Combined with the results in Table 4, which show that the most preferred recommendation mode varied from user to user, these findings indicate that users switched between modes depending on the context. Therefore, it is insufficient to allow users to access only their most-preferred mode. These results highlight the importance of pro-

Table 5. Number of users who actively used k recommendation modes for each interaction type.

Interaction Type	k				Proportion of
	1	2	3	4	users with $k \geq 2$
Song Plays	579	279	155	237	53.68%
Added to Favorites	586	216	96	73	39.65%
Thumbs Up	613	227	116	135	43.81%
Thumbs Down	367	108	46	93	40.23%

viding controllability by enabling users to freely switch between recommendation modes.

6 Conclusion

This paper addressed the development of a recommender system that incorporates both transparency and controllability. Our main contributions can be summarized as follows:

- By extending the existing GCN-based recommendation method, we realized a method that considers various relationships in recommendation and enables the generation of recommendation results emphasizing specific relationships.
- By applying our method, we implemented a recommender system with four switchable recommendation modes and we released it on the music web service Kiite in a publicly accessible form as a case study.
- We analyzed over two years of usage logs from the Kiite recommender system and demonstrated the value of incorporating controllability by allowing users to freely switch between recommendation results based on different recommendation reasons.

Although the layer weight of 100 mentioned in Sect. 4.2 was determined experimentally so that the recommendation results would differ across modes without degrading the recommendation accuracy, the effect on user behavior of using other weight values remains unexplored. Also, in Sect. 5.1, we showed a correlation between the number of different recommendation modes used and the frequency of user interactions with recommended songs, but a causal relationship was not established. While these issues are beyond this paper's scope, we believe they can be examined through A/B testing, and we thus identify them as important topics for future work. Although the current study has limitations regarding these aspects, the academic value of our research remains high, as we have successfully implemented a practical recommender system with transparency and controllability, and we have demonstrated its effectiveness through long-term usage log analysis.

Finally, although Kiite focuses on VOCALOID songs, the included genres are highly diverse, suggesting that our proposed system is effective across a wide range of songs. Furthermore, because our recommendation method can generate recommendations for any data that can be represented as a graph, the same approach can be applied beyond VOCALOID music. We hope that such recommender systems with transparency and controllability will be developed and widely adopted across various music domains.

Acknowledgement. We thank Crypton Future Media, INC. for developing Kiite with us and Nico Nico Douga for initially tacitly (later explicitly) encouraging us to work on this research. We also would like to extend our appreciation to users of Kiite and Nico Nico Douga, creators of VOCALOID songs, and all people who have created, supported, and enjoyed the VOCALOID culture.

This work was supported in part by JST CREST Grant Number JPMJCR20D4 and JSPS KAKENHI Grant Number JP25H01174.

References

1. Balog, K., Radlinski, F., Arakelyan, S.: Transparent, scrutable and explainable user models for personalized recommendation. In: Proceedings of SIGIR '19, pp. 265–274 (2019)
2. Bontempelli, T., Chapus, B., Rigaud, F., Morlon, M., Lorant, M., Salha-Galvan, G.: Flow Moods: recommending music by Moods on Deezer. In: Proceedings of RecSys '22, pp. 452–455 (2022)
3. Bostandjiev, S., O'Donovan, J., Höllerer, T.: TasteWeights: a visual interactive hybrid recommender system. In: Proceedings of RecSys '12, pp. 35–42 (2012)
4. Chatti, M.A., Guesmi, M., Muslim, A.: Visualization for recommendation explainability: a survey and new perspectives. ACM Trans. Interact. Intell. Syst. **14**(3), 1–40 (2024)
5. Dominguez, V., Messina, P., Donoso-Guzmán, I., Parra, D.: The effect of explanations and algorithmic accuracy on visual recommender systems of artistic images. In: Proceedings of IUI '19, pp. 408–416 (2019)
6. Gajdusek, P., Peska, L.: SpotifyGraph: visualisation of user's preferences in music. In: Proceedings of MMM '21, pp. 379–384 (2021)
7. Gedikli, F., Jannach, D., Ge, M.: How should I explain? A comparison of different explanation types for recommender systems. Int. J. Hum Comput Stud. **72**(4), 367–382 (2014)
8. Goto, M.: A chorus section detection method for musical audio signals and its application to a music listening station. IEEE Trans. Audio Speech Lang. Process. **14**(5), 1783–1794 (2006)
9. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of NIPS '17, pp. 1025–1035 (2017)
10. He, X., Chen, T., Kan, M.Y., Chen, X.: TriRank: review-aware explainable recommendation by modeling aspects. In: Proceedings of CIKM '15, pp. 1661–1670 (2015)
11. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: LightGCN: simplifying and powering graph convolution network for recommendation. In: Proceedings of SIGIR '20, pp. 639–648 (2020)
12. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of CSCW '00, pp. 241–250 (2000)
13. Jacobson, K., Murali, V., Newett, E., Whitman, B., Yon, R.: Music personalization at Spotify. In: Proceedings of RecSys '16, p. 373 (2016)
14. Jin, Y., Htun, N.N., Tintarev, N., Verbert, K.: ContextPlay: evaluating user control for context-aware music recommendation. In: Proceedings of UMAP '19, pp. 294–302 (2019)
15. Jin, Y., Seipp, K., Duval, E., Verbert, K.: Go with the flow: effects of transparency and user control on targeted advertising using flow charts. In: Proceedings of AVI '16, pp. 68–75 (2016)
16. Kenmochi, H., Ohshita, H.: VOCALOID - commercial singing synthesizer based on sample concatenation. In: Proceedings of INTERSPEECH '07, pp. 4009–4010 (2007)

17. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of ICLR '15, pp. 1–13 (2015)
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of ICLR '17, pp. 1–14 (2017)
19. Kouki, P., Fakhraei, S., Foulds, J., Eirinaki, M., Getoor, L.: HyPER: a flexible and extensible probabilistic framework for hybrid recommender systems. In: Proceedings of RecSys '15, pp. 99–106 (2015)
20. Kouki, P., Schaffer, J., Pujara, J., O'Donovan, J., Getoor, L.: Personalized explanations for hybrid recommender systems. In: Proceedings of IUI '19, pp. 379–390 (2019)
21. Ma, B., Lu, M., Taniguchi, Y., Konomi, S.: CourseQ: the impact of visual and interactive course recommendation in university environments. Res. Pract. Technol. Enhanc. Learn. **16**(18), 1–24 (2021)
22. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML '13, pp. 1–6 (2013)
23. Millecamp, M., Htun, N.N., Conati, C., Verbert, K.: To explain or not to explain: The effects of personal characteristics when explaining music recommendations. In: Proceedings of IUI '19, pp. 397–407 (2019)
24. Millecamp, M., Htun, N.N., Jin, Y., Verbert, K.: Controlling Spotify recommendations: effects of personal characteristics on music recommender user interfaces. In: Proceedings of UMAP '18, pp. 101–109 (2018)
25. Parra, D., Brusilovsky, P., Trattner, C.: See what you want to see: Visual user-driven approach for hybrid recommendation. In: Proceedings of IUI '14, pp. 235–240 (2014)
26. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of UAI '09, pp. 452–461 (2009)
27. Saito, Y., Itoh, T.: MusiCube: A visual music recommendation system featuring interactive evolutionary computing. In: Proceedings of VINCI '11, pp. 1–6 (2011)
28. Schedl, M.: Deep learning in music recommendation systems. Front. Appl. Math. Statist. **5**, 1–9 (2019)
29. Sinha, R., Swearingen, K.: The role of transparency in recommender systems. In: Proceedings of CHI EA '02, pp. 830–831 (2002)
30. Tsai, C.H., Brusilovsky, P.: Providing control and transparency in a social recommender system for academic conferences. In: Proceedings of UMAP '17, pp. 313–317 (2017)
31. Tsukuda, K., Goto, M.: DualDiv: diversifying items and explanation styles in explainable hybrid recommendation. In: Proceedings of RecSys '19, pp. 398–402 (2019)
32. Tsukuda, K., Goto, M.: Explainable recommendation for repeat consumption. In: Proceedings of RecSys '20, pp. 462–467 (2020)
33. Tsukuda, K., Takahashi, T., Ishida, K., Hamasaki, M., Goto, M.: Kiite world: socializing map-based music exploration through playlist sharing and synchronized listening. In: Proceedings of MMM '25, pp. 197–211 (2025)
34. Vig, J., Sen, S., Riedl, J.: Tagsplanations: explaining recommendations using tags. In: Proceedings of IUI '09, pp. 47–56 (2009)
35. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: KGAT: knowledge graph attention network for recommendation. In: Proceedings of KDD '19, pp. 950–958 (2019)
36. Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: a survey. ACM Comput. Surv. **55**(5), 1–37 (2022)

37. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: Proceedings of ICML '18, pp. 5453–5462 (2018)
38. Zhang, Y., Chen, X.: Explainable recommendation: a survey and new perspectives. Found. Trends Inf. Retr. **14**(1), 1–101 (2020)