A Recommendation Method for an Online Programming Portal

Nguyen Duy Anh[1], Nguyen Duy Phuong[2], Nguyen Van Tien[2]

[1] Department of Cognitive Science and Artificial Intelligence Tilburg University Tilburg, Netherlands
[2] Faculty of Information Technology 1 Posts and Telecommunications Institute of Technology Hanoi, Vietnam

Correspondence: Nguyen Van Tien, tiennv@ptit.edu.vn
Communication: received 28 Aug 2023, revised 6 Oct 2023, accepted 16 Oct 2023
DOI: 10.32913/mic-ict-research.v2024.n1.1227

***Abstract*: An online programming portal is a valuable resource for Information Technology students to enhance their programming skills. It offers an environment where educators and learners can create problems, generate test data, program, and automatically evaluate tests. Numerous universities, both domestic and international, have achieved remarkable success by developing digital content for these portals. However, a common challenge for learners is locating problems that align with their expertise in the vast database covering various topics. In this paper, we propose a collaborative filtering recommendation approach integrated into the Online Programming Portal. This method aims to suggest a suitable set of problems based on each user's programming proficiency. Experiments conducted on the Online Programming Portal at the Post & Telecommunications Institute of Technology (PTIT) demonstrate a significant enhancement in students' online problem-solving results.**

Keywords: *Online Programming Portal Collaborative Filtering Recommendation, Content-based Filtering Recommendation, Item-based Collaborative Filtering Recommendation, User-based Collaborative Filtering Recommendation.*

I. INTRODUCTION TO THE ONLINE PROGRAMMING PORTAL

The computer community is witnessing an unprecedented development of online programming technologies. Online programming technologies provide learners with a multi-language programming environment, allowing them to code and automatically grade their programming results. [1]. Accompanying learners on online programming portals are universities and major economic corporations, all aiming to train and recruit a high-quality IT workforce. Leading IT corporations such as Microsoft, Google, Facebook, Apple, and Samsung, not only establish their online programming portals but also provide financial support and digital content for online programming portals of other universities. [1–3]. Major universities such as MIT, Stanford, and Baylor consider online programming portals as essential tools for teaching, training, and assessing the programming skills of IT engineers. Recognizing the effectiveness of online programming technologies, several universities in Vietnam, including Hanoi National University, Ho Chi Minh City National University, Hanoi University of Science and Technology, and Posts and Telecommunications Institute of Technology, have developed their online programming portals and successfully implemented them in teaching and honing the programming skills of students.

For online programming portals, the most crucial resource is the digital content repository integrated into the platform [1]. This digital content repository consists of a collection of problems, each accompanied by corresponding test datasets. Each problem requires multiple test datasets, with each dataset specifying a correct property that the programming solution must achieve. A programming solution is considered good if it passes all the test datasets with defined time and memory constraints. The digital content repository is constructed and maintained by experts from the organization owning the online programming portal. The larger the digital content repository, the more attractive the online programming platform becomes, leading to a greater number of users' participation [1–3]

The next important resource for an online programming portal is its user community. A large user community indicates the high value of the portal's digital content repository [1]. Most users participate in learning programming, while some join to enhance their programming skills. Others engage in showcasing their programming proficiency and may even compete in national and international programming competitions to validate their expertise and potentially enter the high-quality job market or achieve awards in programming contests at the national and international levels.

Some online programming portals, such as https://codeforces.com/, https://topcoder.com/, and

https://www.hackerrank.com/, attract hundreds of thousands of programmers worldwide to participate, as their digital content is continuously updated. Therefore, building an advisory system to provide tailored digital content and learning progress based on each user's programming abilities is of utmost importance for these online programming portals.

There have been several studies on recommendation systems in online learning environments. Recommendation systems in education have also been explored in recent research trends. In 2019, Carlo De Medio and colleagues conducted research on content recommendation systems for instructors on online learning platforms using the Moodle platform [4]. Asra Khalid and Karsten Lundqvist proposed a new recommendation method for Massive Open Online Courses (MOOCs) in 2021 [5].

Some studies by authors Huynh Ly Thanh Nhan and Nguyen Thai Nghe have focused on predicting students' academic performance using open-source systems [6] or course selection advisory systems [7]. However, these studies may not align well with online programming learning platforms.

In this paper, we propose a collaborative filtering advisory method to enhance the effectiveness of online programming portals. The method is approached by considering the collaborative advisory problem on the online programming portal as a graph-based search problem. Leveraging graph representation, we propose a method to predict the suitability of programming content for each user based on their programming abilities. The method is experimentally evaluated on real-world data collected from the online programming portal of the Posts and Telecommunications Institute of Technology, and it shows promising results compared to traditional collaborative filtering methods. For ease of tracking, in Section 2, we present the collaborative filtering advisory problem for online programming portals. Section 3 introduces the method for collaborative filtering advisory on online programming portals, and the final section presents the experimentation and evaluation of the results.

II. COLLABORATIVE FILTERING ADVISORY PROBLEM FOR ONLINE PROGRAMMING PORTALS

Collaborative filtering recommendation is a popular method in building advisory systems widely applied in e-commerce [8, 9]. The method is built from a set of users $U = \{u_1, u_2, \ldots, u_n\}$ and a set of items $P = \{p_1, p_2, \ldots, p_m\}$. Each user $u_i \in U$ provides their ratings for certain items $p_x \in$ P with a numerical value $r_{ix} \in \Omega$ (where Ω can be a set of integers or a set of real numbers). The rating matrix R is the sole input for collaborative filtering recommendation methods [9]. For ease of presentation, we can use the shorthand notation $i \in U$ and $x \in P$ instead of $u_i \in U$ and $p_x \in P$ respectively. Based on the rating matrix $R = \{r_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\}$, collaborative filtering recommendation methods exploit aspects related to user communities with similar interests to provide the most suitable items for each user. The core philosophy of collaborative filtering is that users with similar preferences in the past may have shared interests in the future. Each user in the collaborative filtering advisory system operates independently from other users.

The problem of content recommendation for each user on the online programming portal can be viewed as a typical collaborative filtering recommendation problem. In this context, the user set $U = \{u_1, u_2, .., u_n\}$ consists of users participating in the online programming portal. The set U is automatically collected as users register to join the online programming platform. The item set $P = \{p_1, p_2, \ldots, p_m\}$ includes problems along with their corresponding test datasets already loaded into the online programming portal. The programming results of each user $i \in U$ solving problem $x \in P$ is automatically recorded by the online programming portal with a rating value r_{ix}. Specifically, r_{ix} is recorded as 1 if the programming solution of user $i \in U$ satisfies all the test datasets for problem $x \in P, r_{ix}$ is recorded as -1 if the programming solution of user $i \in U$ does not fully meet all the test datasets for problem $x \in P$, and r_{ix} is recorded as 0 if the user $i \in U$ has not solved problem $x \in P$. The objective of the collaborative filtering recommendation approach is to offer a collection of problems that match the programming skills of individual users on the online programming portal.

There are various proposals to address the collaborative filtering recommendation problem, and they can be categorized into two main approaches: Memory-Based Collaborative Filtering and Model-Based Collaborative Filtering [8, 9]. In this paper, we focus on the Memory-Based collaborative filtering method. Memory-Based Collaborative Filtering is approached through two primary methods: User-Based Collaborative Filtering [10, 11] and Item-Based Collaborative Filtering [11, 12]. Each method has its own advantages, exploiting aspects related to users or items. A common feature of both methods is that they utilize the entire rating dataset to predict the preferences of users for items they have not encountered before. Essentially, these are lazy learning or example-based learning methods used in machine learning [13]. Each method is carried out through four steps, as described below [10–12].

Step 1. Calculate the similarity between user pairs or item pairs. At this step, we can use correlation measures

or similarity metrics to compute the similarity between user pairs or item pairs [10–12]. Let u_{ij} be the similarity between user $i \in U$ and user $j \in U$, and p_{xy} be the similarity between item $x \in P$ and item $y \in P$. Then, the Pearson correlation between user $i \in U$ and user $j \in U$ is determined by formula (1), and the similarity between item $x \in P$ and item $y \in P$ is determined by formula (2) [10, 12].

$$u_{ij} = \frac{\sum_{x \in P_i \cap P_j} (r_{ix} - \overline{r_l})(r_{jx} - \overline{r_j})}{\sqrt{\sum_{x \in P_i \cap P_j} (r_{ix} - \bar{r}_l)^2} \sqrt{\sum_{x \in P_i \cap P_j} (r_{jx} - \overline{r_j})^2}} \quad (1)$$

$$u_{ij} = \frac{\sum_{x \in P_i \cap P_j} (r_{ix} - \bar{r_i})(r_{jx} - \bar{r_j})}{\sqrt{\sum_{x \in P_i \cap P_j} (r_{ix} - \bar{r_i})^2} \sqrt{\sum_{x \in P_i \cap P_j} (r_{jx} - \bar{r_j})^2}} \quad (2)$$

Where,

$$P_i = \{x \in P \mid r_{ix} \neq 0\} \quad (3)$$

$$\bar{r_i} = \frac{1}{|P_i \cap P_j|} \sum_{x \in P_i \cap P_j} r_{ix} \quad (4)$$

$$\bar{r_j} = \frac{1}{|P_i \cap P_j|} \sum_{x \in P_i \cap P_j} r_{jx} \quad (5)$$

$$U_x = \{i \in U \mid r_{ix} \neq 0\} \quad (6)$$

$$\bar{r_x} = \frac{1}{|U_x \cap U_y|} \sum_{i \in U_x \cap U_y} r_{ix} \quad (7)$$

$$\bar{r_y} = \frac{1}{|U_x \cap U_y|} \sum_{i \in U_x \cap U_y} r_{iy} \quad (8)$$

Step 2. Determine the neighbor set for the user in need of recommendations. At this step, we simply sort the u_{ij} or p_{xy} values in descending order, where $i \in U$ represents the user requiring item recommendations from the set of items $x \in P$. Then, select the first K_i users as the neighbor set for user i, or select the first K_x items as the neighbor set for item x [10, 12].

Step 3. Generate predictions for the user in need of recommendations. The most common method to generate predictions for user $i \in U$ on a new item $x \in P$ is through formula (9), and for items, it is through formula (10) [10–12].

$$r_{ix} = \bar{r_i} + \frac{\sum_{j \in K_i} (r_{jx} - \bar{r_j}) u_{ij}}{\sum_{j \in K_i} |u_{ij}|} \quad (9)$$

$$r_{ix} = \frac{\sum_{y \in K_x} p_{xy} r_{iy}}{\sum_{y \in K_x} |p_{xy}|} \quad (10)$$

Step 4. Formulate the recommendations. Select the top k new items with the largest rix values to recommend to user i, or choose the top k users i with the largest pix values to recommend to these users [10–12].

Despite many successes in collaborative filtering recommendation systems, the User-Based and Item-Based collaborative filtering methods still face some challenges, such as sparse data, new users, and new items [9]. Particularly, collaborative filtering methods heavily depend on the real-world semantics of data values in different application domains. In this paper, we propose a graph-based collaborative filtering method that overcomes these issues. The method proves to be effective even in cases of sparse data.

III. COLLABORATIVE FILTERING RECOMMENDATION ALGORITHM FOR ONLINE PROGRAMMING PORTALS

In order to enhance the quality of recommendations, this section presents a proposed collaborative filtering advisory algorithm for online programming portals. The method is implemented by representing the relationships between users and content items on the online programming portal as a bipartite graph. Leveraging this property, we treat the problem at hand as a graph search problem. The method is carried out as follows.

1. The method estimates the level of suitability of users for items

Let's consider a system with n users U = $U = \{u_1, u_2, .., u_n\}$ and m items $P = \{p_1, p_2, \ldots, p_m\}$. In this context, the user set U is automatically collected as users register to join the online programming portal. The item set P includes problems along with their corresponding test datasets already loaded into the online programming portal. The rating matrix $R = \{r_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\}$ is automatically recorded in the online programming portal using formula (11).

$$r_{ix} = \begin{cases} 1 \text{ if user } i \in U \text{ submits the correct problem } x \in P \\ 0 \text{ if user } i \in U \text{ has not submitted the problem } x \in P \\ -1 \text{ if user } i \in U \text{ submits the incorrect problem } x \in P \end{cases} \quad (11)$$

It is evident that the rating matrix R, defined by formula (11), can be represented as a bipartite graph. One side of the graph comprises the user set U, while the other side includes the problem set P within the online programming portal. The edges of the graph consist of pairs $e = (i, x)$, where $i \in U$ and $x \in P$. There are no edges connecting vertices within the same side. In other words, there are no edges connecting user vertices to other user vertices, and there are no edges connecting item vertices to other item vertices. For instance, in a system with a rating matrix comprising 5 users and 7 items, as shown in Table 1, the corresponding bipartite graph representation is depicted in Figure 1. Here, the value $r_{ix} = 1$ indicates that user i has solved problem x correctly, $r_{ix} = -1$ indicates that user i

TABLE I
THE RATING MATRIX R

Users	Items						
	p1	p2	p3	p4	p5	p6	p7
u1	1	0	-1	1	0	-1	0
u2	0	1	-1	1	-1	0	-1
u3	-1	1	1	-1	0	0	1
u4	-1	-1	0	0	1	-1	1
u5	0	1	0	-1	1	1	0

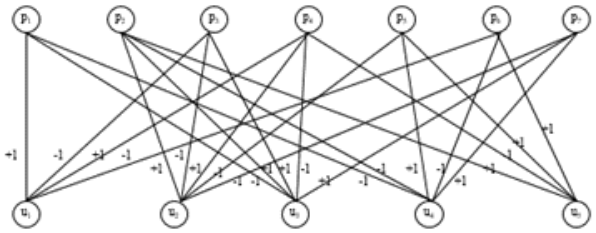

Figure 1. Bipartite graph corresponding to R

has not solved problem x correctly, and $r_{\text{ix}} = 0$ indicates that user i has not attempted problem x yet.

According to the property of the bipartite graph, the paths from the vertex of user $i \in U$ to the vertex of item $x \in P$ always have odd lengths ($L = 1, 3, 5, 7 \cdots$). For example, the paths $u1-p4-u3-p2, u1-p3-u2-p7, u1-$ $p1-u2-p2$ are paths of length 3 , and the paths $u1-p4-$ u2-p2-u3-p7, u2-p3-u1-p6-u4-p1, u1-p1-u3-p7-u4-p2 are paths of length 5 . All paths from the vertex of user $i \in U$ to the vertex of item $x \in P$ can be classified into three types: type 1 includes paths that only pass through edges with positive weight (+1), type 2 includes paths that only pass through edges with negative weight (-1), and type 3 includes paths that pass through edges with either positive (+1) or negative (-1) weights. For example, $u1-p4-u3-p2, u1-$ $p4-u2-p2-u3-p7$ are type 1 paths; $u1-p3-u2-p7, u2-p3-u1-p6-u4-p1$ are type 2 paths; $u1-p1-u2-p2, u1-p1-u3-$ $p7-u4-p2$ are type 3 paths.

By visual observation, if the majority of the paths of length L from vertex $i \in U$ to vertex $x \in P$ belong to type 1 among all three types of paths, then the prediction that item x is suitable for user i is most likely correct. In this case, we predict the opinion of user i about the new item x to be $r_{\text{ix}} = 1$. If the majority of the paths of length L from vertex $i \in U$ to vertex $x \in P$ belong to type 2 among all three types of paths, then the prediction that item x is not suitable for user i is most likely correct. In this case, we predict the opinion of user i about the new item x to be $r_{\text{ix}} = -1$. In the final case, if the majority of the paths of length L from vertex $i \in U$ to vertex $x \in P$ belong to type 3 among all three types of paths, then we cannot predict whether the item x is suitable or not for user i. In this case, we predict the opinion of user i about the new item x to be $r_{\text{ix}} = 0$. Based on this observation, we propose the method to calculate the suitability of users for new items as follows.

Let $W = \{w_{ix} : i = 1, 2, .., n; x = 1, 2, \ldots, m\}$ be the adjacency matrix representing the bipartite graph of the rating matrix R, defined by formula (12). $W^1 = \left\{w^1_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\right\}$ is the adjacency matrix representing the bipartite graph for positive rating values, determined by the formula (13). $W^2 = \left\{w^2_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\right\}$ is the adjacency matrix representing the bipartite graph for negative rating values, determined by the formula (14). In other words, we divide the bipartite graph representation of the rating matrix R into two subgraphs, where subgraph W^l only includes edges with positive weights (+1), and subgraph W^2 only includes edges with negative weights (−1).

$$W = \begin{cases} 1 & \text{if } r_{ix} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$W^1 = \begin{cases} 1 & \text{if } r_{ix} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$W^2 = \begin{cases} 1 & \text{if } r_{ix} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

In that case, the total number of all paths with length L from vertex $i \in U$ to vertex $x \in P$ in the entire rating matrix R is determined by the formula (15) [14]. This is also the count of all three types of paths from vertex $i \in U$ to vertex $x \in P$. The number of type 1 paths (paths that pass through edges with weight 1) from vertex $i \in U$ to vertex $x \in P$ is determined by the formula (16) [14]. The number of type 2 paths (paths that pass through edges with weight -1) from vertex $i \in U$ to vertex $x \in P$ is determined by the formula (17) [14]. The number of type 3 paths (paths that pass through edges with weight 1 or -1) from vertex $i \in U$ to vertex $x \in P$ is determined by formula (18) [14]. In these formulas, W^T is the transpose matrix of W, $\left(W^1\right)^T$ is the transpose matrix of W^1, and $\left(W^2\right)^T$ is the transpose matrix of W^2.

$$W^L = \begin{cases} W & \text{if } L = 1 \\ W \cdot W^T \cdot W^{L-2} & \text{if } L = 3, 5, \ldots \end{cases} \tag{15}$$

$$\left(W^1\right)^L = \begin{cases} W^1 & \text{if } L = 1 \\ W^1 \cdot \left(W^1\right)^T \cdot \left(W^1\right)^{L-2} & \text{if } L = 3, 5, \ldots \end{cases} \tag{16}$$

$$\left(W^2\right)^L = \begin{cases} W^2 & \text{if } L = 1 \\ W^1 \cdot \left(W^2\right)^T \cdot \left(W^1\right)^{L-2} & \text{if } L = 3, 5, \ldots \end{cases} \tag{17}$$

$$\left(W^3\right)^L = W^L - \left(W^1\right)^L - \left(W^2\right)^L \quad (18)$$

So, we have determined the number of each type of path with length L from vertex $i \in U$ to vertex $x \in P$. Let's call MAX the number of paths with length L that is the largest from vertex $i \in U$ to vertex $x \in P$, which is determined by the formula (19). Then, the method to predict the level of suitability for user $i \in U$ with new items $x \in P$ is determined by the formula (20).

$$MAX = max\left\{w_{ix}^L : i = 1,\ 2,\ ..,n; x = 1,\ 2,\ ..,m\right\} \quad (19)$$

$$r_{ix} = \left\{ \begin{array}{ll} 1 & \text{if } \frac{(w^1)_{ix}^L}{MAX} > 0.5 \\ 0 & \text{if } \frac{(w^3)_{ix}^L}{MAX} > 0.5 \\ -1 & \text{if } \frac{(w^2)_{ix}^L}{MAX} > 0.5 \end{array} \right\} \quad (20)$$

In the prediction formula (20), to limit the pairs (i, x) that have a small number of paths with length L but have a larger number of paths with length L belonging to each type compared to w_{ix}^L, we compare it with the maximum value in the matrix W^L for comparison. The predicted value of user i 's opinion for the new item x is $r_{ix} = 1$ when the ratio $\frac{(w^1)_{ix}^L}{MAX} > 0.5$. This means that the number of paths with length L from vertex i to vertex x must be sufficiently large, and the majority of these paths must pass through edges with positive weights.

Similarly, the predicted value $r_{ix} = 1$ when the number of paths with length L from vertex i to vertex x must be sufficiently large, and the majority of these paths must pass through edges with negative weights. The predicted value $r_{ix} = 0$ when the number of paths with length L from vertex i to vertex x must be sufficiently large, and the majority of these paths must pass through edges with both positive and negative weights. Based on the prediction method developed according to (20), we propose a collaborative filtering recommendation algorithm for the online programming portal as in Section III.2.

2. The Collaborative Filtering Recommendation Algorithm for Online Programming Portal

The Collaborative Filtering Recommendation Algorithm for Online Programming Portal (GraphBased) is performed through four steps as shown in Algorithm 1. In Step 1 of the algorithm, we construct bipartite graphs. The graph W is used to determine the number of paths with length L from vertex $i \in U$ to vertex $x \in P$. The graph W^1 is used to determine the number of paths with length L from vertex $i \in U$ to vertex $x \in P$ that only traverse edges with positive weights. The graph W^2 is used to determine the number of paths with length L from vertex $i \in U$ to vertex $x \in P$ that only traverse edges with negative weights.

In Step 2 of the algorithm, we find the number of paths with length L from vertex $i \in U$ to vertex $x \in P$ in the graph W. This allows us to observe indirect relationships between

Algorithm 1 The GraphBased algorithm.

Inputs: The rating matrix R is determined according to the formula (11).
Outputs: The list of k most suitable new items $p_x \in P$ for the user $p_i \in U$.
The steps are as follows:
Step 1. Constructing bipartite graphs from the rating matrix R:
1.1. Build the adjacency matrix representing the bipartite graph over the entire R according to the formula (12): $W = \begin{cases} 1 & \text{if } r_{ix} \neq 0 \\ 0 & \text{otherwise} \end{cases}$
1.2. Build the adjacency matrix representing the bipartite graph for ratings with a value of 1 using the formula (13):

$$W^1 = \left\{ \begin{array}{ll} 1 & \text{if } r_{ix} > 0 \\ 0 & \text{otherwise} \end{array} \right.$$

1.3. Build the adjacency matrix representing the bipartite graph for ratings with a value of -1 using the formula (14): $W^2 = \begin{cases} 1 & \text{if } r_{ix} < 0 \\ 0 & \text{otherwise} \end{cases}$
Step 2. Finding the number of paths of length L for each type:
2.1. Find the number of paths of length L from vertex $i \in U$ to vertex $x \in P$ over the entire R using formula (15): $W^L = \begin{cases} W & \text{if } L = 1 \\ W \cdot W^T \cdot W^{L-2} & \text{if } L = 3, 5, \ldots \end{cases}$
2.1. Find the number of type 1 paths of length L from vertex $i \in U$ to vertex $x \in P$ using formula (16):

$$\left(W^1\right)^L = \left\{ \begin{array}{l} W^1 \\ W^1 \cdot \left(W^1\right)^T \cdot \left(W^1\right)^{L-2} \quad \text{if } L = 3, 5, \ldots \end{array} \right.$$

2.2. Find the number of type 2 paths of length L from vertex $i \in U$ to vertex $x \in P$ using formula (17):

$$\left(W^2\right)^L = \left\{ \begin{array}{ll} W^2 & \text{if } L = 1 \\ W^1 \cdot \left(W^2\right)^T \cdot \left(W^1\right)^{L-2} & \text{if } L = 3, 5, \ldots \end{array} \right.$$

2.3. Find the number of type 3 paths of length L from vertex $i \in U$ to vertex x ∈ P using formula (18):

$$\left(W^3\right)^L = W^L - \left(W^1\right)^L - \left(W^2\right)^L$$

Step 3. Predicting the opinions of $i \in U$ regarding the new items $x \in P$:
3.1. Find the number of paths of length L from vertex $i \in U$ to vertex $x \in P$ over the entire R using formula (15): $MAX = \max\left\{w_{ix}^L : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\right\}$
3.2. Generate predictions for the opinions of $i \in U$ regarding the new items $x \in P$:

$$r_{ix} = \left\{ \begin{array}{ll} 1 & \text{if } \frac{(w^1)_{ix}^L}{MAX} > 0.5 \\ 0 & \text{if } \frac{(w^3)_{ix}^L}{MAX} > 0.5 \\ -1 & \text{if } \frac{(w^2)_{ix}^L}{MAX} > 0.5 \end{array} \right.$$

Step 4. Providing recommendations for user $i \in U$ for the new items $x \in P$:
4.1. Sort r_{ix} in ascending order of weights.
4.2. Select the first k new items with $r_{ix} = 1$ to recommend to user i.

user pairs in determining similarity. The result obtained is matrix W^{L}, which records the number of all types of paths with length L from vertex $i \in U$ to vertex $x \in P$. Next, we find the number of paths with length L that traverse edges with positive weights in the matrix $\left(W^{1}\right)^{L}$, and the number of paths with length L that traverse edges with negative weights in the matrix $\left(W^{1}\right)^{L}$. Finally, we determine matrix $\left(W^{1}\right)^{L}$ which records the number of paths with length L that traverse edges with both positive and negative weights.

In Step 3 of the algorithm, we proceed to fill in the predicted values rix according to the formula (20). In this step, some labels initially having $r_{\mathrm{ix}} = 0$ are replaced with the value 1 , while others are replaced with the value -1 . The remaining parts have a value of 0 , corresponding to the situation where we cannot make a prediction. The detailed algorithm is illustrated in Algorithm 1.

IV. EXPERIMENTATION AND EVALUATION

The GraphBased algorithm proposal was experimentally conducted on a dataset collected from the online programming portal of the Posts and Telecommunications Institute of Technology. The process of constructing the dataset and the experimental results were evaluated and compared with other methods according to the procedure described below.

1. Experimentation Methodology

Firstly, the entire test data is divided into two parts, one part U_{tr} is used as the training data, and the remaining part U_{te} is used for testing. The U_{tr} set contains 80% of the ratings, and the U_{te} set contains 20% of the ratings. The training data is used to build the model using the algorithm described above. For each user i in the test dataset, their ratings (already available) are split into two parts: O_{i} and P_{i}.O_{i} is considered as the known data, while P_{i} is the ratings to be predicted using the training data and O_{i} [9, 10, 12].

The prediction error MAE_{u} for each customer u in the test dataset is calculated as the mean absolute error between the predicted values and the actual values for all items in the set P_{u}.

$$MAE_u = \frac{1}{|P_u|} \sum_{y \in P_u} \left| \hat{r}_{uy} - r_{uy} \right| \tag{21}$$

$$MAE = \frac{\sum_{u \in U_{te}} MAE_u}{|U_{te}|} \tag{22}$$

2. Test Dataset

The collaborative filtering method was directly collected by the research team from the online programming portal of the Posts and Telecommunications Institute of Technology. The dataset was collected from August 2020 to June 2023, with 6,435 users registered on the online programming portal. The repository of programming tasks was built within one year, consisting of 1,245 problems for learners to program and be automatically graded. The total number of problem submissions recorded in the online programming portal until June 2023 was 1,151,000, submitted by 1,435,000 users. For each submission, if user i correctly solved problem x, the portal recorded a value of 1. If user i attempted but did not solve problem x, the portal recorded a value of -1. If user i did not attempt problem x at all, the portal recorded a value of 0. Each user could submit a problem multiple times, but the system only recorded the final result with values 1 or -1. Out of the 6,435 users, we filtered down to 6,120 users who participated in programming at least 20 problems correctly or incorrectly for testing purposes.

We randomly selected 2,000, 3,000, and 4,000 users from the 6,120 users as the training data and randomly selected 400, 600, and 800 users from the remaining users as the test data. To evaluate the performance of the proposed method compared to other methods in cases with limited data, we adjusted the number of problems programmed by each user in the test data to be 5, 10, and 20, with the remaining submissions for prediction.

3. Comparison and Evaluation

The proposed GraphBased method in Section III was tested and compared with the following methods:

- UserBased method using Pearson correlation, which is a user-based collaborative filtering method presented in Section II [10, 11].

- ItemBased method using Pearson correlation, which is an item-based collaborative filtering method presented in Section II [11, 12].

The MAE values in Table 2 were estimated from the average of 10 random experiments. The test results showed that the proposed method consistently yielded lower MAE values than the UserBased and ItemBased methods in all cases where the known data was 5, 10, or 20 evaluations. Specifically, in the case of very sparse data with 5 known evaluations in the training dataset of 2000 users, the MAE values for the UserBased, ItemBased, and GraphBased methods were 0.2158, 0.2172, and 0.208, respectively. The MAE values of the UserBased, ItemBased, and GraphBased methods tended to decrease as the size of the training dataset increased to 3000 and 4000 users. However, the GraphBased method still outperformed the other methods, indicating its effectiveness even in sparse collaborative filtering scenarios.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT RECOMMENDER SYSTEMS.

Training dataset size	Method	Number of known ratings/reviews		
		5	10	20
2000 users	UserBased	0.2158	0.2117	0.2042
	ItemBased	0.2172	0.2146	0.1992
	GraphBased	**0.2068**	**0.1984**	**0.1872**
3000 users	UserBased	0.2147	0.2012	0.1924
	ItemBased	0.2145	0.2116	0.1967
	GraphBased	**0.2043**	**0.1877**	**0.1792**
4000 users	UserBased	0.2037	0.2117	0.1942
	ItemBased	0.2012	0.2146	0.1820
	GraphBased	**0.1987**	**0.1784**	**0.1712**

In the case of relatively complete data, specifically with 20 known evaluations, the MAE value of the GraphBased method is significantly lower than the other methods. The MAE values of the GraphBased method are 0.1812, 0.1792, and 0.1712, while both the UserBased and ItemBased methods yield MAE values > 1.820. This can be explained by the fact that the user-based and item-based collaborative filtering methods directly estimate the degree of similarity between user-item pairs based on the intersection of item or user evaluations. However, with the graph-based method, we can infer the degree of similarity over sets of paths with positive weights and sets of paths with negative weights while not making predictions with paths containing both positive and negative weights. This allows us to leverage indirect relationships in the prediction results.

The number of students participating in programming learning and skill development has doubled within 4 months after implementing the collaborative filtering recommendation method into the online programming portal. The number of student submissions has increased from 2000 per week to 8000 per week. The average online practice time for each student is 98 hours, compared to 8 hours of offline practice according to the teaching program. These results further affirm the important role of the online programming portal in enhancing the programming skills of learners. The recommendation method established on the online programming portal is an effective tool that supports users in finding suitable problems based on their programming abilities.

V. CONCLUSION

The paper has proposed a collaborative filtering recommendation method on the online programming portal of the Posts and Telecommunications Institute of Technology to provide each user with a set of problems suitable for their programming abilities. The proposed model has proven effective even in cases of sparse data, which is a common challenge for other collaborative filtering recommendation models. The research team has also constructed a relatively comprehensive collaborative filtering dataset to share with the research community interested in using it. The proposed collaborative filtering method has brought practical benefits to online programming learners at the Posts and Telecommunications Institute of Technology.

In addition to the collaborative filtering dataset, we are currently working on building datasets for content-based recommendations based on user profiles, problem topic information, user-submitted programs, and user comments on solutions to problems on the online programming portal. This will serve as valuable data for implementing content-based recommendation methods, hybrid recommendation methods and experimenting with other recommendation approaches.

REFERENCES

[1] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Comput. Surv. (CSUR)*, vol. 51, pp. 1–34, 2018.

[2] J. Liu, S. Zhang, Z. Yang, Z. Zhang, J. Wang, and X. Xing, "Online judge system topic classification," 2018, pp. 993–1000.

[3] P. Antonucci, C. Estler, D. Nikolić, M. Piccioni, and B. Meyer, "An incremental hint system for automated programming assignments," 06 2015, pp. 320–325.

[4] C. Medio, C. Limongelli, F. Sciarrone, and M. Temperini, "Moodlerec: A recommendation system for creating courses using the moodle e-learning platform," *Computers in Human Behavior*, vol. 104, 03 2020.

[5] A. Khalid, K. Lundqvist, A. Yates, and M. a. Ghazanfar, "Novel online recommendation algorithm for massive open online courses (nor-moocs)," *PLOS ONE*, vol. 16, p. e0245485, 01 2021.

[6] H.-L. Thanh-Nhan and N. Thai-Nghe, "A system for predicting students's course result using a free recommender system library - mymedialite," 11 2013.

[7] D. Tran Thanh, L. Sang, N. Thanh Hai, and N. Thai-Nghe, "Course recommendation with deep learning approach," *Communications in Computer and Information Science*, vol. 1306, pp. 63–77, 11 2020.

[8] F. Tawfiq, A. M. Rahma, and H. Abdul wahab, "Recommendation systems for e-commerce systems an overview," *Journal of Physics: Conference Series*, vol. 1897, p. 012024, 05 2021.

[9] N. P. Raval and V. Khedkar, "A review paper on collaborative filtering based moive recommedation system," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:220844886

[10] Z. Zhang, T. Peng, and K. Shen, "Overview of collaborative filtering recommendation algorithms," *IOP Conference Series: Earth and Environmental Science*, vol. 440, p. 022063, 03 2020.

[11] P. Thakkar, K. Varma (Thakkar), V. Ukani, S. Mankad, and S. Tanwar, *Combining User-Based and Item-Based Collaborative Filtering Using Machine Learning: Proceedings of ICTIS 2018, Volume 2*, 01 2019, pp. 173–180.

[12] M. Kharita, A. Kumar, and P. Singh, "Item-based collaborative filtering in movie recommendation in real time," 12 2018, pp. 340–342.
[13] B.-B. Cui, "Design and implementation of movie recommendation system based on knn collaborative filtering algorithm," *ITM Web of Conferences*, vol. 12, p. 04008, 01 2017.
[14] T. Phuong, D. T. Lien, and N. D. Phuong, "Graph-based context-aware collaborative filtering," *Expert Systems with Applications*, vol. 126, 02 2019.

Nguyen Duy Anh born on September 26, 2001, is currently a fourth-year student majoring in Artificial Intelligence at Tilburg University in the Netherlands. His research interests include machine learning, artificial intelligence, and information filtering.
Email: ndanh@gmail.com

Nguyen Duy Phuong born on February 20, 1965 in Hanoi. Graduated from Hanoi University of Science in 1998, defended a doctoral thesis at the University of Engineering and Technology, Hanoi National University in 2010. Currently working at the Post and Telecommunications Institute of Technology. Research focus: Machine Learning Applications in Information Filtering, Expert Systems, and Artificial Intelligence.
Email: ndaphuong@gmail.com

Nguyen Van Tien received a Bachelor's degree in Information Technology in 2018 and a Master's degree in Information Systems in 2020 from the Posts and Telecommunications Institute of Technology. From June 2020 to the present, Mr. Nguyen Van Tien has been a lecturer at the Faculty of Information Technology 1, Posts and Telecommunications Institute of Technology. His main research directions include high-performance computing systems and deep learning-based recommendation systems
Email: nvtien@gmail.com