New Explainable Overlapping Co-Clustering for Recommender Systems: Capturing Multifaceted Preferences with Enhanced Interpretability

Chiheb Eddine Ben Ncir, Mohammed Ibrahim Alattas
College of Business, University of Jeddah, Saudi Arabia

***Abstract*—Recommender systems have become critical tools in reducing information overload by providing personalized recommendations across several application domains including commerce, industry, education, academic research, etc. Clustering-based recommender systems, which use the clustering technique to group similar users or items to generate suggestions, have shown high accuracy and efficiency. However, conventional clustering methods often fail to address several challenges such as ignoring the possibility that a user may have different item preferences, limited interpretability of generated suggestions, and the inability to tailor recommendation list sizes to individual user needs. To address all these issues, we propose in this work a new recommender system based on Overlapping Co-clustering and Modularity Maximisation (OCCMM). The proposed method allows to take into account that users may have several item preferences by building overlapping clusters rather than the conventional non-overlapping model. Also, the proposed method adopts a simultaneous clustering of items and users to facilitate the generation and interpretation of suggestions through using the co-clustering technique. Furthermore, OCCMM enables an adjustment of recommendation list sizes based on an easy tunning parameter δ. Experiments conducted in three real-world datasets demonstrated the effectiveness of OCCMM in achieving better performance in terms of accuracy and interpretability compared to conventional existing methods.**

***Keywords*—Clustering-based recommender systems; modularity maximization; overlapping co-clustering; multiple-user-preferences; recommendation interpretability**

I. Introduction

Recommender systems play a vital role in addressing the challenge of information overload and providing personalized recommendations in various domains. These systems have become valuable tools for companies to deliver targeted information to users or customers. The implementation of a high-performance recommender system is usually based on various machine-learning methods and techniques such as clustering which allows for the grouping of similar users or items into the same cluster. The generation of recommendations is based on similar items or users belonging to the same cluster. Such Clustering-based recommender systems are usually characterized by an improved running time with high accuracy of recommendations [1].

Typically, clustering methods used in recommender systems have focused on either user-based or item-based recommendations. User-based recommendations focus on similarities between users where recommendations are generated based on the preferences and behavior of similar users. This recommendation approach is relatively simple to implement as it only requires user-item interaction data. However, it faces challenges when dealing with new users who have limited data or when recommending items with limited user feedback, which leads to less accurate recommendations. On the other side, item-based recommendations focus on similar items rather than users and tend to be more stable as item preferences are less likely to change compared to user preferences. Item-based recommendations allow to effectively handle new items since similar item characteristics can be used to recommend them. However, this approach may lead to less personalized recommendations since it focuses on item-item similarities and ignores user preferences.

To take advantage of both user-based and item-based approaches, the co-clustering technique can be used rather than using the clustering technique. The Co-clustering allows for simultaneous grouping of users and items allowing recommendations to be based on both characteristics simultaneously [2] [3]. However, most existing co-clustering methods assume non-overlapping co-clusters and do not allow overlaps between groups. Nevertheless, in real-life applications, users may have multiple preferences and items can satisfy multiple user needs resulting in overlapping co-clusters. Overlapping co-clustering can yield more accurate recommendation results by capturing all user's preferences and item needs. Additionally, a shortcoming with existing clustering-based recommendation methods is the absence of options for regulating the size of the recommendation list. This lack of control hampers the flexibility of the recommendation system, making it challenging to customize the number of recommendations according to user preferences.

To address all these challenges, this paper proposes a dual contribution at both the theoretical and practical levels. From a theoretical perspective, it advances the literature on recommender systems by introducing a novel formulation of the Overlapping Co-Clustering with Modular Maximization (OCCMM) framework. Unlike existing works, the proposed model integrates 1) an enhanced algorithmic formulation that optimizes modularity while supporting overlapping user–item clusters, 2) an explicit interpretability mechanism that allows practitioners to understand why certain recommendations are generated, and 3) a tunable list sizing strategy that adapts dynamically to users' contextual needs. From a practical perspective, the framework proposes explainable recommendations in domains where transparency is critical, such as e-commerce, tourism, and educational platforms. We will show the ability of the proposed OCCMM model not only to generate explanations that are transferable and generalizable to other application areas.

The rest of the paper is organized as follows: Section II presents the fundamental concepts of recommender systems. Section III discusses the challenges addressed by the proposed approach. Section IV provides a detailed theoretical framework for the overlapping co-clustering method based on modularity maximization. Section V presents the experimental results and evaluates the performance of our approach using real-world datasets. Finally, Section VI concludes the paper and outlines directions for future research.

II. Fundamental Concepts of Recommender Systems

With the exponential growth of available information across databases, web pages, and social networks, the task of extracting valuable information for users has become increasingly challenging. Users often struggle to quickly and easily find items of interest in this huge amount of available data. To address this issue, automatic recommendation tools, known as Recommender Systems (RS), have been developed. The primary objective of these systems is to minimize the time of searching and providing personalized suggestions. By analyzing extensive amounts of data through a variety of techniques and algorithms [4], these computing-based systems try to generate personalized recommendations for products or services based on individual user preferences [5]. These recommendations can span a wide range of categories, such as articles to read, products to purchase, music to listen to, movies to watch, or web pages to visit.

The effectiveness of recommendations and suggestions mainly rely on the system input data types which can be categorized into two main types: Explicit and Implicit Feedback Data. Explicit feedback data refers to information provided by users in a clear and detailed manner, typically in the form of ratings given to a set of products. This type of data directly expresses the user's preferences. On the other hand, implicit feedback data is automatically collected based on user behavior and then transformed into user preferences. Examples of implicit feedback data include clicks on links, browsing history, the number of times a song is played, or the percentage of a web page scrolled. These data are collected, converted into user preferences, and then utilized to generate recommendations.

The collected data are often represented on a scale to indicate the level of interest in a particular item. It can take various forms, such as continuous values, as seen in the Jester joke recommendation engine [6], where jokes are rated on a scale from -10 to 10. Data can also be modeled using intervals, where the lowest value represents a strong dislike and the highest indicates a strong liking for an item. Another approach is to model ratings as ordered categorical values, enabling the extraction of all possible user ratings. The commonly used set of categories includes "Strongly Disagree", "Disagree", "Neutral", "Agree" and "Strongly Agree". Ratings can also be represented as binary data, where users express their preferences as either liking or disliking an item. Additionally, a specific data type known as unary rating data can be used in recommender systems, where users only have the option to define a like for an item without the ability to specify a disliking. All these different types of data play a vital role in building accurate and effective recommender systems, allowing for the generation of personalized recommendations tailored to individual user preferences.

A. Recommender System Approaches

The definition and the building of recommender systems can be based on several approaches. Three main approaches can be identified: *Collaborative filtering*, *Content-based filtering*, and *Knowledge-based filtering*. The first approach, collaborative filtering, operates on the assumption that users with similar interests may have the same item preferences. Collaborative filtering methods can be roughly divided into two categories : *User-based* and *Item-based* collaborative filtering. The User-based collaborative filtering identifies an ensemble of users who are similar to a target user (i.e. n neighbors). Based on the neighbor's ratings, it predicts the possible rating(s) to a list of products not yet purchased (i.e. product(s) to be recommended) or the top N-recommended products. In this technique, similarity functions are calculated between rows of rating matrix to extract similar users [4]. The second collaborative filtering category, Item-based, tries to extract the highly correlated items (similar items) to the ones that target users are interested in. The extracted items are considered as recommendations. In this category, similarity functions are computed between the columns of the rating matrix, rather than rows, to discover similar items [4].

Concerning the second recommender systems approach, *Content-based filtering*, both user ratings and item properties are used to build recommendations. In this approach, the similarity is calculated based on item features (i.e. product content). Recommended items consist of the filtered list of similar products to those a target user shows high interest. Such type of filter does not suppose that other users must make recommendations for the target user. Based only on what the target user likes, the algorithm will simply pick items with similar contents and recommend them to that user. A content-based filtering system recommends items based on the correlation between the content of the items. This approach is different from the collaborative filtering approach which recommends items based on the correlation between users/items with similar preferences [7].

The third recommender systems approach, knowledge-based, aims to provide recommendations based on product properties, issued by the user, instead of using the history of users' preferences. This approach allows users to explicitly express their desires or explicitly indicate their item preferences before the generation of recommendations. A recommender system is considered "knowledge-based" when it prompts the user to give a series of rules or guidelines on what the results should look like or an example of an item. Then, the system searches through its database of items and returns similar results [8].

B. Related Works: Clustering-Based Recommender Systems

Clustering has been used as a significant technique in the design and the implementation of scalable and efficient recommender systems, known as clustering-based recommender systems. Such systems are based on clustering algorithms to group users or items based on shared preferences or behaviors. The built segments of items or users allow for the recommendation system to focus on specific group characteristics

to improve the computational efficiency and to address data sparsity challenges and cold start problem [9]. For instance, the authors in [10], [11] and [12] used clustering to efficiently recommend news articles based on user interests. The author in [13] leveraged clustering in smart grid applications to enhance recommendations. Ahuja et al. [14] built a Movie recommender system using K-Means clustering and K-Nearest Neighbor. The authors in [15], [16] and [17] utilized clustering in tourism recommendation to group users with similar travel interests.

Recent advancements in clustering-based recommendation systems explore various clustering algorithms and approaches to optimize system performance, user satisfaction, and computational efficiency. Existing methods can be grouped into three main approaches: user-based, item-based, and hybrid.

The first, user-based, groups users with similar behaviors in the same cluster and then recommendations are generated based on local preferences within each cluster. For instance, Yu et al. [18] proposed ClusterFedMet [18], a recommendation algorithm that addresses the challenges of recommendations in federated learning environments. ClusterFedMet improves recommendation accuracy and personalization while keeping data privacy. Additionally, Hosen et al. [19] proposed an optimized recommender system, using PSO-optimized K-means clustering, able to create patient-specific diet recommendations based on historical health data. The proposed system ensures rapid and relevant dietary suggestions for thyroid patients.

On the other hand, the second approach, item-based, allows to group similar items having shared attributes or similar user-interactions[20]. Zhang and Wu [21] developed an e-commerce recommendation model that integrates K-means with a genetic algorithm to enhance recommendation accuracy and speed. The proposed model has a significant boost in item relevance and user satisfaction compared to existing ones. Rather than using a simple clustering method, Airen and Agrawal [22] applied the co-clustering technique in movie recommendations. The fine-tuning of user and movie neighborhood parameters allowed to increase recommendation quality. Their model showed notable improvements in accuracy by leveraging a partitioned clustering approach within large movie datasets.

Concerning the third approach, hybrid clustering, it combines both user and item clustering to simultaneously capture user preferences and item characteristics, leading to a comprehensive recommendation strategy with improved suggestions accuracy and relevance. A recent example of hybrid systems proposed by Wang et al. [23] for tourism recommendation combines multiple recommendation methods, including collaborative filtering and neural networks, to suggest different and relevant tourist spots. A multi-objective approach is used to suggest recommendations that allows to enhance users travel experience. Another hybrid approach, proposed by Iftikhar et al. [24] use bi-clustering within a reinforcement learning framework to adapt recommendations to changing user preferences. This model allows to capture local patterns in user-item interactions by combining bi-clustering with Markov Decision Processes which makes it effective for dynamic recommendation scenarios. Furthermore, another system proposed by Forouzandeh et al. [25], referred to as RESCHet, embeds spectral clustering in heterogeneous networks to analyze relationships in complex item-user networks. The use of spectral clustering to group item-user interactions made it ideal for large datasets where traditional clustering methods may fall short.

All these presented approaches were based on common Clustering algorithms including k-means, hierarchical clustering, density based and Gaussian Mixture Models. The choice of the clustering technique and also the choice of the similarity metric are usually based data characteristics and application requirements. For example, when dealing with sparse datasets, k-means++ or DBSCAN may be favored due to their robustness against sparse data [26]. A good choice of these techniques has a direct impact on the accuracy of the system. Although the effective application of clustering techniques in recommendation systems, conventional techniques may oversimplify the complexity of user preferences by assigning each user or item to a single cluster. However, in real life applications, a more flexible approach is mandatory since users or items may have several preferences and then may belong to multiple clusters. For example, when recommending movies, a user may have interests in both “romantic” and “action” genres of films. This reflects the multifaceted nature of user preferences and the necessity to use more sophisticated and flexible clustering models.

As a solution for the multifaceted nature of user preference, Overlapping clustering methods have proven to be particularly useful in capturing these diverse preferences [27], [28], [20]. Recent works illustrate the advantages of overlapping clustering in various domains. For instance, Heckel and Vlachos [27] introduced a matrix factorization-based approach to identify overlapping co-clusters of clients and products. This method emphasizes interpretability by recommending items that are members of multiple co-clusters. Effectiveness is shown especially in large datasets making it ideal for business-to-business applications. Besides, Vlachos et al. [28] expand the overlapping co-clustering concept to address cold-start problems in matrix factorization. Flexibility of recommendations is improved by allowing users and items to participate in multiple clusters. This flexibility ensures that new users or items are matched effectively which avoids the limitations of traditional clustering methods. Rezghi and Baratnezhad [2] also proposed fuzzy co-clustering methods using non-negative matrix factorization. Overlapping clustering is used to improve the quality of recommendations by reflecting users’ diverse preferences. The proposed approach achieved better sparcity reduction and accuracy compared to non-overlapping methods. Iftikhar et al. [24] also employed overlapping co-clustering within a reinforcement learning framework. By allowing users and items to belong to multiple bi-clusters, the proposed model allowed to effectively capture local patterns and dynamic preferences compared to static collaborative filtering methods, especially in dynamic environments where user preferences may evolve over time. Furthermore, the authors in [29] explored fairness improvement in recommender systems using overlapping clustering. They propose grouping users based on interactions and sensitive attributes, such as age or gender, to provide equitable recommendations across diverse user groups. Overlapping clustering have lead to more balanced outcomes and improved fairness metrics. In the same way, the authors [30] leveraged overlapping clustering to optimize recommendation diversity and novelty. By allowing overlaps, users can explore new items while maintaining access to

familiar favorites. This design has significantly enhanced the diversity of recommendations without compromising precision.

III. Motivation and Research Gap

The design of effective clustering-based recommender systems remains a major research challenge given that these systems must simultaneously ensure accuracy, diversity, interpretability, and computational efficiency. While numerous clustering methods have been proposed, their application to recommender systems still suffer from fundamental limitations that restrict their theoretical and practical contributions.

A first challenge relates to the treatment of user and item clusters as mutually exclusive. Existing clustering-based approaches assume that a user or an item belongs to a single homogeneous group. However, in real-world applications, user preferences are often heterogeneous and multidimensional. For instance, a movie enthusiast may simultaneously enjoy action, drama, and comedy genres, while a book reader may have overlapping interests in historical fiction, biographies, and mystery. Ignoring such overlaps leads to rigid partitions that fail to capture the nuanced and overlapping structures of user preferences. An overlapping co-clustering approach, by contrast, can represent these multifaceted relationships more faithfully and can lead to more personalized recommendations.

A second critical issue concerns interpretability. Recommender systems users usually demand greater transparency regarding the rationale behind generated recommendations. Existing clustering-based methods typically optimize predictive accuracy but fall short in providing explainable justifications for their outputs, particularly when overlapping user–item relationships are involved. Co-clustering offers a promising solution by jointly segmenting users and items. Co-clustering makes it possible to explain recommendations in terms of shared membership across multiple co-clusters which directly improves trust, accountability, and user acceptance.

A third methodological gap concerns the tailoring of recommendation list sizes. Current systems frequently adopt fixed-length recommendation lists, overlooking the dynamic and context-dependent nature of user needs. This "one-size-fits-all" strategy reduces the relevance of recommendations. It becomes possible to adapt list sizes dynamically by integrating the degree of overlap in user–item co-clusters. Such flexibility enables the system to remain context-aware and user-centric, thereby enhancing recommendation utility.

Despite recent advances, existing approaches still exhibit significant limitations. Conventional hard-partition methods remain insufficient to capture multi-faceted user preferences, while soft-clustering approaches provide smoother decision boundaries but lack the structured dual segmentation of co-clustering. Moreover, current methods rarely balance the trade-off between interpretability and predictive performance; most prioritize one dimension at the expense of the other. Finally, although prior studies have begun to explore dynamic list adaptation [2], [30], there is no comprehensive framework that unifies overlap-based co-clustering with modularity maximization, interpretability mechanisms, and personalized list sizing.

To address these gaps, this paper proposes an Overlapping Co-Clustering approach with Modularity Maximization. The framework contributes novel theoretical and practical advancements by: 1) capturing complex overlapping structures of user–item interactions, 2) integrating interpretability into the clustering process to support explainable recommendations, and 3) dynamically adapting list sizes to user contexts. This approach directly responds to the discussed challenges and establishes a more accurate, interpretable, and adaptable recommender system.

IV. Proposed Method: Recommendation based on Overlapping Co-Clustering and Modularity Maximization

The proposed method, called Overlapping Co-clustering with Modularity Maximization (OCCMM), is designed to provide overlapping co-clusters based on modularity maximization using implicit feedback data. The overlapping co-clusters will be then explored to generate flexible and easily interpretable recommendations. The proposed OCCMM method begins by representing the data of the recommender system as a bipartite graph, where users and items are represented as distinct types of vertices. The edges of the graph correspond to the positive feedback given by users on the items. To illustrate the designed OCCMM recommendation process, we have schematized the main steps in Fig. 1. Our proposed method consists of three key steps. First, it begins by building a binary matrix from the bipartite graph. Next, it focuses on creating an ensemble of co-clusters, which will be utilized in the final step to generate recommendations. In the following subsections, we provide a detailed breakdown of the tasks involved in each step of the OCCMM process.

A. Step 1: Data Preprocessing

The data preprocessing step has the objective of effectively transforming the input bipartite graphs of the recommender system into a binary adjacency matrix. Let $G = (N, M, E)$ be the input bipartite graph, where N represents the set of user nodes, M the set of item nodes and E represents the set of edges. The objective is to transform the bipartite Graph G into a binary adjacency matrix $D = (N \times M)$. The value $d_{nm} = 1$ indicates the existence of a relationship (edge in the graph G) between the user node n and the item node m. The adjacency matrix D encompasses all relationships between users and items, serving as implicit feedback data that will be automatically segmented into k overlapping co-clusters to detect similarities between items and users simultaneously. Fig. 1 shows an example of transforming a bipartite graph containing $N = 6$ users and $M = 4$ items into an adjacency matrix $D = (6 \times 4)$ where entries $d_{nm} = 1$ represents the existence of an edge between user n and item m. For instance, the edge between $user_1$ ($n = 1$) and $item_1$ ($m = 1$) indicates that $user_1$ bought $item_1$, prefers it, or any other relevant connection between $user_1$ and $item_1$ resulting in $d_{11} = 1$.

B. Step 2: Generation of Co-Clusters

Given the implicit feedback data matrix $D = (N \times M)$ obtained from the previous step, the objective of the second step is to look for optimal k co-clusters of users $U(N \times k)$ and items $I(M \times k)$ simultaneously based on an iterative modularity-maximization-based process. As described in the

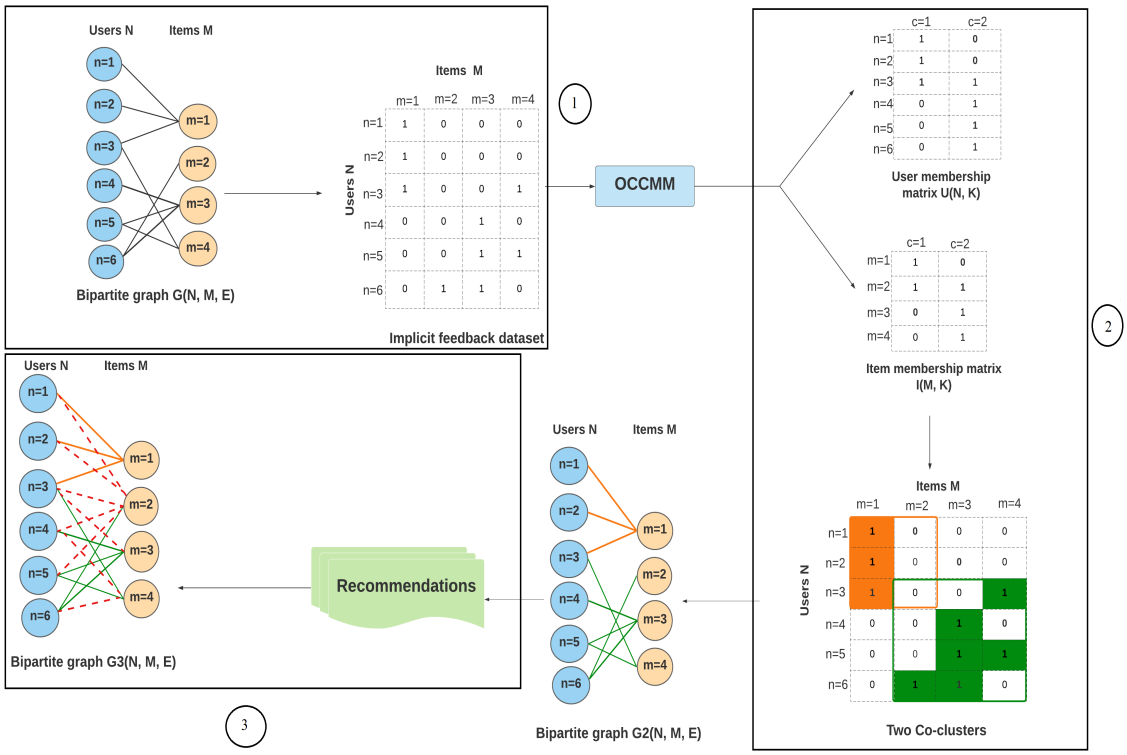

Fig. 1. The designed recommendations steps via the proposed OCCMM method. Step 1 : Data preprocessing, Step 2: generation of co-clusters, Step 3: Generating recommendations using Co-clusters.

works of [27] that uses a modularity maximization of a network for building optimal co-clusters, we define the following modularity objective function for the feedback data:

$$Q(U,I) = \frac{1}{|D|}\sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{c=1}^{k}(d_{nm} - \frac{d_n d_m}{|D|})u_{nc}i_{mc} \quad (1)$$
$$-\sum_{n=1}^{N}\sum_{c=1}^{k}u_{nc}\log u_{nc} - \sum_{m=1}^{M}\sum_{c=1}^{k}i_{mc}\log i_{mc}$$

where, $|D| = \sum_n \sum_m d_{nm}$ represents the total number of edges in the network, $d_{nm} \in \{0,1\}$ is an element of the feedback matrix equal to one when a positive feedback of the user n on item m exists, $d_n = \sum_{m=1}^{M} d_{nm}$ and $d_m = \sum_{n=1}^{N} d_{nm}$ represent the degree of user node n and the degree of item node m respectively. The values $u_{nc} \in [0,1]$ and $i_{mc} \in [0,1]$ indicate the fuzzy membership degrees that user n and item m belong to cluster c, respectively. The values $\sum_{n=1}^{N}\sum_{c=1}^{k} u_{nc}\log u_{nc}$ and $\sum_{m=1}^{M}\sum_{c=1}^{k} i_{mc}\log i_{mc}$ denote the separate entropy regularizing users and items membership degree functions separately. The minimization of these two values corresponds to maximizing the fuzzy entropy $-\sum_{n=1}^{N}\sum_{c=1}^{k} u_{nc}\log u_{nc}$ and $-\sum_{m=1}^{M}\sum_{c=1}^{k} i_{mc}\log i_{mc}$.

The maximization problem of the objective function Q respecting to optimal matrices (U) and (I) can be described follows:

max

$$Q(U,I) \quad (2)$$

subject to

$$\sum_{c=1}^{N} u_{nc} = 1, \quad u_{nc} \in [0,1], \quad \forall n = 1,\ldots,N \quad (3)$$
$$\sum_{c=1}^{M} i_{mc} = 1, \quad i_{mc} \in [0,1], \quad \forall m = 1,\ldots,M \quad (4)$$

This problem can be solved through the application of the Lagrange method, involving the utilization of Lagrange multipliers ζ_a and η_b corresponding to constraints (3) and (4) respectively. Eq. (1) can be expressed as:

$$\begin{aligned} Q(U,I) =& \frac{1}{|D|}\sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{c=1}^{k}\left(d_{nm} - \frac{d_n d_m}{|D|}\right)u_{nc}i_{mc} \\ &- \sum_{n=1}^{N}\sum_{c=1}^{k}u_{nc}\log u_{nc} - \sum_{m=1}^{M}\sum_{c=1}^{k}i_{mc}\log i_{mc} \\ &+ \sum_{n=1}^{N}\zeta_n\left(\sum_{c=1}^{k}u_{nc} - 1\right) + \sum_{m=1}^{M}\eta_m\left(\sum_{c=1}^{k}i_{mc} - 1\right) \end{aligned} \quad (5)$$

By deriving (Q) with respect to (U) and (I) and setting the gradient to zero, the following expressions are obtained:

$$\frac{\partial Q}{\partial U} = \sum_{m=1}^{M} \left(d_{nm} - \frac{d_n d_m}{|D|} \right) i_{mc} - (1 + \log u_{nc}) + \zeta_n = 0$$
$$\iff$$
$$u_{nc} = \frac{\exp\left(\sum_{m=1}^{M} \left(d_{nm} - \frac{d_n d_m}{|D|} \right) i_{mc} \right)}{\sum_{c=1}^{k} \exp\left(\sum_{m=1}^{M} \left(d_{nm} - \frac{d_n d_m}{|D|} \right) i_{mc} \right)} \tag{6}$$

$$\frac{\partial Q}{\partial I} = \sum_{n=1}^{N} \left(d_{nm} - \frac{d_n d_m}{|D|} \right) u_{nc} - (1 + \log i_{mc}) + \eta_m = 0$$
$$\iff$$
$$i_{mc} = \frac{\exp\left(\sum_{n=1}^{N} \left(d_{nm} - \frac{d_n d_m}{|D|} \right) u_{nc} \right)}{\sum_{c=1}^{k} \exp\left(\sum_{n=1}^{N} \left(d_{nm} - \frac{d_n d_m}{|D|} \right) u_{nc} \right)} \tag{7}$$

Eq. (6) and (7) refer to the fuzzy membership of user n to co-cluster c and the fuzzy membership of item m to co-cluster c, respectively. When the membership degrees of user n and item m surpass a threshold θ defined in Eq. (8), u_{nc} and i_{mc} are set to 1, signaling the membership of user n and item m to co-cluster c.

$$\theta = \left(\frac{1}{k} + \frac{1}{\delta \times k} \right) \tag{8}$$

where, k denotes the number of co-clusters and $\delta \in]1, +\infty[$ represents a user-defined parameter introducing flexibility in recommendations. A higher value of δ yields a more extensive list of recommendations, while a lower value restricts the recommendations.

To look for optimal overlapping co-clusters that maximize the overall modularity function, the optimization process starts with random initializations of matrices U and I and an initial modularity score Q_0. Subsequently, a series of operations are iterated by computing new optimal values for user cluster memberships u_{nc} and item cluster memberships i_{mc} using Eq. (6) and (7), respectively. Following this, a new modularity score $Q_t(U, I)$ at iteration t is recalculated based on these updated memberships. This iterative procedure continues until the modularity score remains unchanged or a maximum iteration limit is reached, offering a systematic approach to organizing users and items into fuzzy co-clusters. The final overlapping co-clusters are then built by assigning users and items to similar co-clusters if their membership probabilities surpass a predefined threshold θ. A pseudo-code of the Overlapping Co-clustering via Modularity Maximization algorithm is described in Algorithm 1.

In the context of clustering, modularity allows to measure how well a network is divided into clusters by comparing the actual connections to a randomly connected network. In our context, we aim to find co-clusters of users and items that maximize a certain modularity that ensures users in the same cluster interact with similar items more frequently than a randomly expected one. Based on the adjacency matrix $D(N \times M)$ of users U and items I, the modularity function $Q(U, I)$ evaluates how well users and items are grouped into overlapping co-clusters while maintaining statistical significance. The term $d_{nm} - \frac{d_n d_m}{|D|}$ captures the difference between the actual interactions and the expected random interactions, ensuring that clusters a better meaningful relationships. Entropy terms (log functions) encourage fuzzy membership, indicating that users and items can belong to multiple co-clusters rather than a strict one-to-one assignment.

Algorithm 1 Overlapping Co-clustering via Modularity Maximization

Input: D= Original data matrix, k= number of co-cluster, δ: recommendation flexibility parameter, IterationLimit: Number of maximal iterations

Output: Co-clusters matrices U and I

```
Random initialization of U and I
repeat
  Compute u_nc according to Formula (6)
  Compute i_mc according to Formula (7)
  Compute Q(U, I) using Formula (1) based on u_nc and i_mc
until Unchanged Q(U,I) or IterationLimit
for n = 1 to N do
  for c = 1 to k do
    if u_nc > θ then
      Assign the user n to the co-cluster c
    end if
  end for
end for
for m = 1 to M do
  for c = 1 to k do
    if i_mc > θ then
      Assign the item m to the co-cluster c
    end if
  end for
end for
End
```

C. Step 3: Generating Recommendations using Co-Clusters

The third step is devoted to building recommendations based on the obtained overlapping co-clusters. This step is based on the hypothesis that users and items within the same co-cluster exhibit a strong correlation. By leveraging the relationships identified in the co-clusters, the algorithm generates personalized recommendations for users based on items they have not yet purchased but are correlated within the same co-cluster. An item becomes a potential recommendation for a user when both the user and the item are grouped in the same co-cluster but lack a direct link (e.g. a purchase).

The pseudo-code of generating recommendations using overlapping co-clustering is described in Algorithm 2. Given the input data matrix D and the overlapping co-clusters U and I, the recommendation algorithm examines each user-item pair in the dataset. In cases where the user has not interacted with the item ($d_{nm} = 0$), the algorithm checks if that user and that item belongs to the same co-cluster. If this condition is satisfied, the item is included in the recommendation list for that user.

In Fig. 1, colored squares within co-clusters represent positive or implicit examples, while white squares inside the co-clusters signify recommendations. Analogous to graph theory, the task consists in predicting probable edges in the graph $G(N, M, E)$ for near nodes. The predicted edges, shown as red dotted lines in the $G(N, M, E)$ graph, correspond to user recommendations.

Algorithm 2 Recommendation using Overlapping Co-Clusters

Input: D: Original data matrix, U, I: Overlapping Co-clusters
Output: Users recommendation lists

```
for n = 1 to N do
  for m = 1 to M do
    if d_nm=0 then
      {The user n has not yet purchased the item m}
      for c = 1 to k do
        if u_nc = 1 and i_mc = 1 then
          {Item m and user n are in the same co-cluster}
                        n-RecommendedList ← m
        end if
      end for
    end if
  end for
end for
End
```

D. Computational Effectiveness

The OCCMM computational complexity can be evaluated through the computational complexity of its three main steps: data preprocessing, generation of overlapping co-clusters, and recommendation generation. The computational complexity of the first step, data preprocessing, involves constructing a binary adjacency matrix from the bipartite graph $G(N, M, E)$ and requires iterating over the edges. The computational complexity of this step is approximated by $O(E)$. The second step, co-cluster generation, is the most computationally consuming as it involves iteratively optimizing the user-item matrix based on the proposed modularity-maximization process. The computational complexity of the modularity function requires $O(NM)$ operations per iteration, and updating the membership matrices involves summing over all users and items, leading to a complexity of $O(kNM)$ per iteration, where k is the number of co-clusters. Given that this process is repeated until convergence with a maximal iteration number of t, the total computational complexity of the second step can be evaluated by $O(TkNM)$. Concerning the third step, recommendation generation, involves checking all user-item pairs to determine potential recommendations based on co-cluster memberships, which is approximated by $O(kNM)$. Summing all the computational costs across the three steps, the overall OCCMM computational complexity is evaluated by $O(TkNM)$. In practical applications, this computational complexity can be considered efficient. In real-world applications, the number of iterations T required for convergence is generally small, and the number of co-clusters k is significantly smaller than N and M, which largely reduces excessive computational demands. Consequently, OCCMM can be approximated by $O(NM)$ making it effective even with large-scale recommender systems.

V. Experiments and Empirical Results

A. Datasets Description

To assess the effectiveness of our method, we employed three real-world datasets, each offering distinct types of implicit feedback. The first dataset, from the CiteULike website[1], contains user-curated collections of scientific articles. This dataset comprises $5,551$ users and $16,980$ articles, and the goal is to generate new article recommendations based on these positive examples. In this dataset, recommendations are only based on binary implicit feedback, where each user's collection of articles is treated as positive examples. The data is represented as a binary matrix $D = N \times M$, where $d_{nm} = 1$ indicates that user n has added article m to their collection, and $d_{nm} = 0$ indicates the absence of that article.

The second dataset is a subset of the Last.fm dataset[2], which captures listeners' preferences for artists in a binary form. This dataset consists of $1,226$ listeners and 285 artists and the recommendation task consists of recommending likely preferred artists for each listener based on their implicit feedback. In this dataset, the input matrix $D = N \times M$ has $d_{nm} = 1$ if listener n has shown interest in artist m, and $d_{nm} = 0$ otherwise.

The third dataset is the Movielens dataset which contains $100,000$ movie ratings from 671 users on $9,066$ movies. Users rate movies on a scale of 1 to 5 stars, and we follow the convention from previous research [31], [32] by considering ratings of 3 or higher as positive feedback and ignoring lower ratings. This dataset is also transformed into a binary matrix, with ratings of 3 or above taking a value of 1. The task is to generate movie recommendations based on these positive ratings. Across all datasets, the recommendation problem is modeled as generating new suggestions using implicit feedback, represented consistently in binary matrix form.

Table I summarizes the description of used datasets by giving the number of users, the number of items, and the recommendation task for each dataset.

TABLE I. Datasets Description

Datasets	Users	Items	Recommendation Objective
CiteUlike	5551	16980	Article recommendation
LastFm	1226	285	Music recommendation
Movielens	671	9066	Movie recommendation

The selected datasets were chosen to validate different aspects of our model, including its robustness, ability to model complex user preferences, and performance across varying levels of data density. The CiteULike dataset was chosen for its high sparsity making it a challenging scenario to test OCCMM's ability to handle sparse academic interactions. CiteULike contains approximately 5500 users, 17000 articles,

[1]Launched in October 2004 at the University of Manchester and closed permanently on March 30, 2019.

[2]https://www.last.fm/

and 200000 interactions. The LastFm dataset, containing 1900 users, 17600 artists, and 92800 interactions, was selected due to its representation of diverse musical preferences and multi-genre listening behaviors, allowing for assessing OCCMM's capability to capture overlapping user interests. Finally, we included the Movielens dataset, a relatively denser dataset that enables us to evaluate the scalability and generalizability of OCCMM in less sparse settings.

B. Evaluation Methodology

We assessed the efficacy of the recommendations by measuring Recall and Precision [33]. Fig. 2 schematizes the different steps to calculate Precision and Recall measures to evaluate generated recommendations. The dataset is split into training and testing sets by removing a portion of interactions for each user, simulating unseen data. For instance, with a test set ratio of 0.2, if a user had 10 interactions, 8 were allocated to training, and 2 were reserved for testing. Subsequently, the training data was fed into the OCCMM algorithm to form co-clusters. These co-clusters were then used to generate recommendations. Finally, the test set was utilized to evaluate the recommendation outcomes based on Precision and Recall metrics.

The Precision metric quantifies the ratio of relevant items among the retrieved items, aiming to gauge the proportion of recommended items that are pertinent. It is calculated in Eq. (9) as:

$$Precision\ (n) = \frac{|\{m : d_{nm} = 1\} \cap \{m_1, ..., m_Z\}|}{\{m_1, ..., m_Z\}} \quad (9)$$

where: $|\{m : d_{nm} = 1\}|$ denotes the list of relevant items, $|\{m_1, ..., m_Z\}|$ represents the list of recommended items, and $|\{m : d_{nm} = 1\} \cap \{m_1, ..., m_Z\}|$ represents the list of recommended items that are relevant.

Then, the Recall measure assesses the fraction of relevant items that have been retrieved out of the total relevant items. It measures the proportion of relevant items recommended by the system and is calculated in Eq. (10) as:

$$Recall\ (n) = \frac{|\{m : d_{nm} = 1\} \cap \{m_1, ..., m_Z\}|}{|\{m : d_{nm} = 1\}|} \quad (10)$$

To provide a single balanced score, we build the F_Measure, described in Eq. (11), by computing the harmonic mean of Precision and Recall as follows:

$$F\_Measure(n) = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (11)$$

As for precision and Recall, a higher F_Measure value indicates a superior recommendation system. The maximum attainable value for the F_Measure is 1, reflecting perfect precision and recall alignment, while the minimum score is 0, indicating a lack of precision and recall balance.

Furthermore, we assess the Average Recommendation metric (Eq. (12)) that calculates the percentage of recommendations provided for each user, crucial for evaluating recommendations generated from overlapping co-clusters. A high recommendation rate is considered when this value exceeds 50%, indicating that each user receives over half of the recommendations and is computed as:

$$Average\_Recommendation = \frac{\sum_{n=1}^{N}\{m_1, ..., m_z\}}{N} \times \frac{1}{M} \quad (12)$$

where, N and M are respectively the total numbers of users and items and $\sum_{n=1}^{N}\{m_1, ..., m_z\}$ represents the total number of recommended items.

C. Number of Co-Clusters and Number of Iterations Parameters

The quality of co-clusters and the convergence of the OCCMM method relies significantly on the pre-determined parameters : size of co-clusters K and the number of iterations. A successful determination of these two parameters can lead to enhanced recommendation quality [27]. Concerning the choice of the optimal number of co-clusters K, we determined this value across various evaluations of co-clusters and modularity values. Fig. 3 gives the obtained scores of modularity by considering different values of co-clusters evaluated on all the three datasets. The plot illustrates distinct modularity patterns across the three datasets (CiteUlike, LastFm, and Movielens) as the number of co-clusters varies. For CiteUlike, the modularity peaks at 0.570 with 22 co-clusters, but it declines beyond this point which suggests overfitting as additional co-clusters are introduced. LastFm shows a more gradual trend, with a peak modularity of 0.312 obtained with 16 co-clusters. Movielens peaks at 0.398 with only 5 co-clusters suggesting that this dataset has fewer distinct clusters. Its modularity decreases rapidly as more co-clusters are added which indicates that the dataset may have a simpler structure and is more sensitive to over-segmentation. All these obtained scores show that all three datasets exhibit a clear peak modularity followed by a decline, reflecting a tendency toward overfitting when the number of co-clusters becomes too large. However, the variation in peak modularity and the number of co-clusters across the datasets highlights differences in their inherent complexity. CiteUlike requires a higher number of co-clusters to capture its structure, whereas Movielens has a much simpler grouping, as indicated by its peak with only 5 co-clusters. This emphasizes the importance of dataset-specific tuning when applying co-clustering methods to maximize modularity.

As for co-clusters, we also studied the influence of the number of iterations parameters in the convergence of the proposed OCCMM method. This study involved evaluating OCCMM modularity across varying number of iterations for CiteUlike, Lastfm, and Movielens datasets. Fig. 4 illustrates the convergence behavior of OCCMM for each dataset over multiple number of iterations. For CiteUlike, the modularity starts close to zero and rapidly increases in the first 10 iterations, with minimal further improvements beyond 10 iterations. For LastFm, the modularity starts similarly near zero but increases more gradually than CiteUlike, eventually

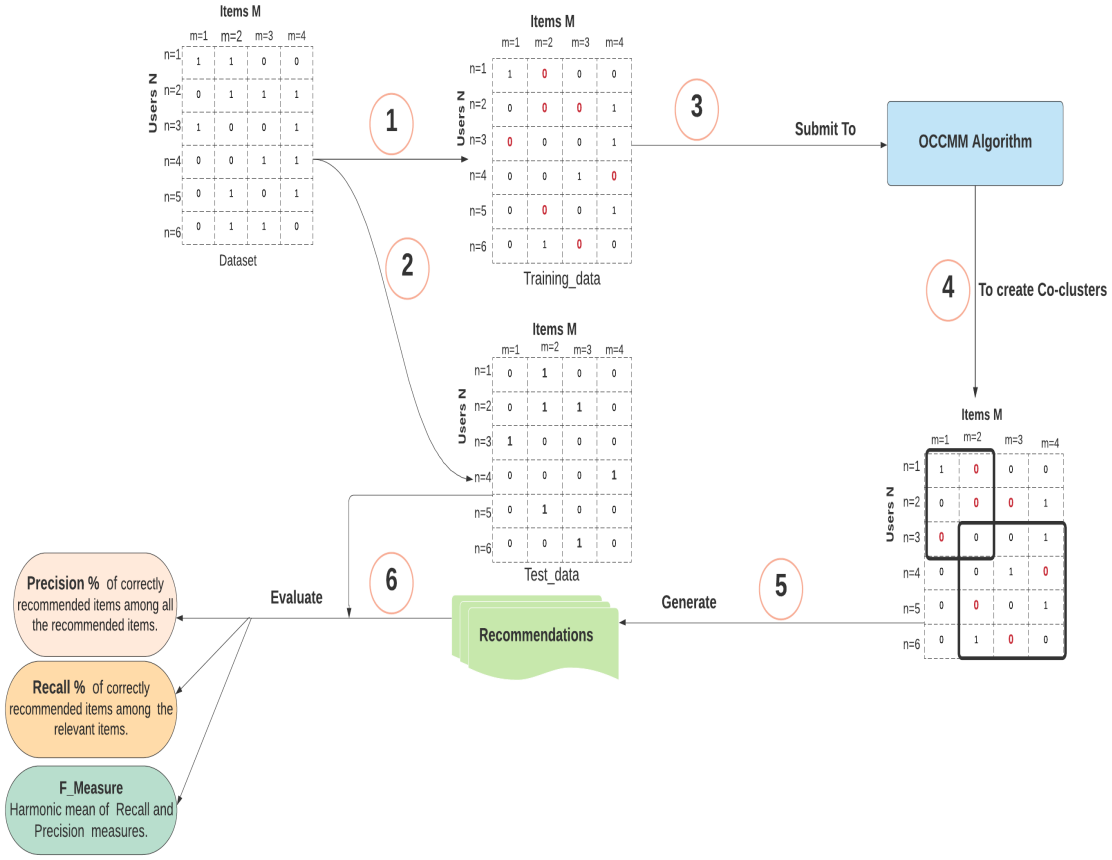

Fig. 2. Evaluation methodology and calculation of evaluation measure based on training and testing datasets.

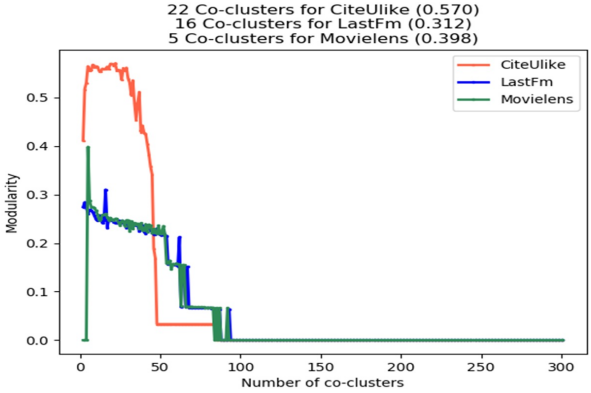

Fig. 3. Maximum modularity values for the CiteUlike, LastFm, and Movielens datasets obtained with OCCMM across varying numbers of co-clusters. The figure shows the optimal number of co-clusters identified for each dataset, along with the corresponding maximum modularity values.

stabilizing around a modularity of 0.3 after 15 iterations. This behavior indicates that OCCMM method finds a good clustering solution earlier with fewer defined or less well-separated clusters compared to CiteUlike, and that additional iterations do not improve the modularity. For Movielens, the modularity rises to about 0.4 after around 18 iterations to be stabilized. This indicates that the dataset has a clear clustering structure that OCCMM quickly identifies good partitionnings, but the co-clustering quality is not as high as in CiteUlike. Across all datasets, we observe that most of the improvement in modularity occurs within the first 10 to 15 iterations, after which the algorithm stabilizes with minimal further gains. For this reason, we set the iterationLimit parameter to 20 for all the further described experiments.

VI. Obtained Empirical Results

We evaluated the effectiveness of the proposed *OCCMM* method in building effective recommendations compared to existing methods in the literature based on *precision*, *recall*, *FMeasure*, and *Average_Recommendation* metrics. Tables II, III, and IV report the obtained values for these measures across different datasets and various methods, including *Bayesian Personalized Ranking (BPR)*, *Fuzzy Co-clustering Modularity Maximization (MMFCC)*, and *Co-clustering via Modularity Maximization (COCLUS)*. These methods are compared with a varying number of co-clusters: $K = 5$, $K = 10$, $K = 16$, and $K = 22$. For our proposed *OCCMM* method, we report results for different values of δ, specifically $\delta = 1$, $\delta = 10^2$, and $\delta = 10^3$.

Concerning CiteUlike dataset, obtained results in Table II show that the proposed **OCCMM** method consistently

TABLE II. OBTAINED PRECISION, RECALL, AND FMEASURE OF OCCMM OF BPR VS MMFCC VS COCLUS VS OCCMM, CITEULIKE DATASET

Datasets	Algorithms	δ value	Precision	Recall	F_Measure	Avg_Recommendation
$K = 5$	BPR		0.014	0.054	0.022	0.058
	MMFCC		0.180	0.755	0.290	0.214
	COCLUS		0.116	0.709	0.199	0.199
	OCCMM	**1**	0.193	0.754	**0.307**	0.212
		10^2	0.171	0.802	0.281	0.243
		10^6	0.171	0.802	0.281	0.244
$K = 10$	BPR		0.014	0.056	0.022	0.058
	MMFCC		0.141	0.683	0.233	0.123
	COCLUS		0.135	0.632	0.222	0.129
	OCCMM	**1**	0.141	0.720	**0.235**	0.136
		10^2	0.134	0.777	0.228	0.172
		10^6	0.132	0.778	0.225	0.178
$K = 16$	BPR		0.014	0.057	0.022	0.058
	MMFCC		0.112	0.664	0.191	0.110
	COCLUS		0.129	0.630	0.214	0.106
	OCCMM	**1**	0.149	0.722	**0.247**	0.130
		10^2	0.136	0.785	0.231	0.170
		10^6	0.135	0.786	0.230	0.171
$K = 22$	BPR		0.014	0.058	0.022	0.058
	MMFCC		0.110	0.713	0.190	0.128
	COCLUS		0.117	0.559	0.193	0.067
	OCCMM	**1**	0.201	0.761	**0.318**	0.138
		10^2	0.195	0.826	0.315	0.188
		10^6	0.162	0.827	0.270	0.189

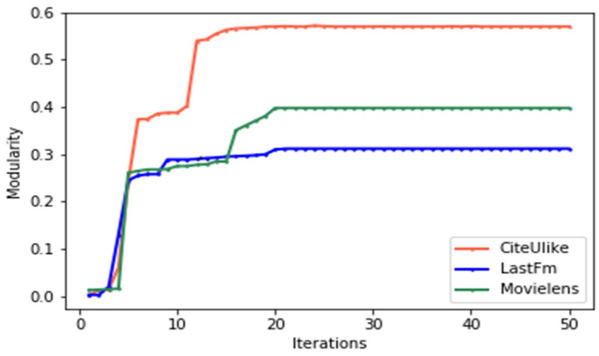

Fig. 4. Maximum modularity values for the CiteUlike, LastFm, and Movielens datasets obtained with OCCMM across varying numbers of iterations. The figure illustrates how the modularity stabilizes after a certain number of iterations for each dataset.

achieves the highest FMeasure values across all K values when $\delta = 1$, highlighting its effectiveness in generating accurate recommendations. For instance, with $K = 22$ and $\delta = 1$, OCCMM achieves an FMeasure of 0.318, largely outperforming both MMFCC and COCLUS having an FMeasure values of 0.190 and 0.193 respectively. This result demonstrates OCCMM superior capability in balancing precision and recall. In terms of recall, OCCMM also performs strongly, especially with larger δ values. With $K = 22$ and $\delta = 10^6$, OCCMM achieves a recall of 0.827, meaning that 82.7% of relevant items are successfully recommended—significantly higher than the recall rates of the other methods. The BPR method achives the loest result eit a recall around 0.058 across all K values, indicating that only 5.8% of relevant items are recommended. However, overlapping methods including MMFCC and COCLUS reach better recall values close to 0.7 meaning that these methods can recommend approximately 70% of relevant items.

Precision values generally range from 0.10 to 0.20 for all methods except BPR, suggesting that, on average, about 10–20 $\%$ of recommendations are relevant to the user. Although OCCMM's precision is comparable to MMFCC and COCLUS, its superior recall and FMeasure values indicate a better overall balance, especially with smaller δ values. The δ parameter in OCCMM also allows for customization of the recommendation list size. For example, with $K = 22$ and $\delta = 1$, OCCMM yields an AverageRecommendation value of 0.138. However, if users prefer a larger recommendation list, setting δ to 10^6 increases the AverageRecommendation to around 0.2, providing more options at the cost of a slight reduction in precision.

For the second dataset, LastFm, the reported results in Table III highlight the competitive performance of OCCMM compared to BPR, MMFCC, and COCLUS in terms of FMeasure and recall metrics across different co-cluster sizes ($K = 5$, $K = 10$, $K = 16$, and $K = 22$). OCCMM achieves the highest FMeasure scores (0.365 and 0.401) for $K = 10$ and $K = 16$ with $\delta = 10^2$. This improvement in FMeasure is driven by a significant increase in recall. OCCMM reaches its highest recall values of 0.628 and 0.729 for $K = 16$ and $K = 22$ with $\delta = 10^2$, indicating that OCCMM can recommend around 70% of relevant items. Additionally, the results demonstrate OCCMM's flexibility in adjusting the recommendation list size based on the configured δ parameter. As δ increases, OCCMM provides larger Average_Recommendation values, reflecting its ability to increase the overlap degree and consequently enlarge

TABLE III. EVALUATION METRICS RESULTS OF BPR VS MMFCC VS COCLUS VS OCCMM, LASTFM DATASET

Datasets	Algorithms	δ value	Precision	Recall	F_Measure	Avg_Recommendation
$K = 5$	BPR		0.012	0.242	0.160	0.175
	MMFCC		0.275	0.481	0.349	0.184
	COCLUS		0.227	0.485	0.309	0.194
	OCCMM	1	0.265	0.457	0.335	0.160
		10^2	0.267	0.556	**0.360**	0.236
		10^6	0.267	0.556	**0.360**	0.237
$K = 10$	BPR		0.013	0.254	0.024	0.175
	MMFCC		0.132	0.413	0.200	0.083
	COCLUS		0.134	0.315	0.188	0.096
	OCCMM	1	0.289	0.440	0.348	0.152
		10^2	0.277	0.535	**0.365**	0.222
		10^6	0.277	0.536	**0.365**	0.223
$K = 16$	BPR		0.013	0.286	0.024	0.175
	MMFCC		0.210	0.552	0.304	0.015
	COCLUS		0.134	0.362	0.195	0.118
	OCCMM	1	0.247	0.398	0.304	0.185
		10^2	0.295	0.626	**0.401**	0.333
		10^6	0.282	0.638	0.391	0.345
$K = 22$	BPR		0.014	0.315	0.026	0.175
	MMFCC		0.163	0.727	**0.266**	0.459
	COCLUS		0.141	0.244	0.178	0.074
	OCCMM	**1**	0.163	0.727	**0.266**	0.459
		10^2	0.163	0.727	**0.266**	0.459
		10^6	0.116	0.759	0.201	0.493

the recommended list of items.

Concerning the Movielens dataset, the results in Table IV illustrate the performance of BPR, MMFCC, COCLUS, and OCCMM across different co-cluster sizes ($K = 5$, $K = 10$, $K = 16$, and $K = 22$). OCCMM consistently demonstrates strong performance in terms of FMeasure, especially for higher values of δ. For instance, OCCMM achieves the highest FMeasure scores of 0.288 and 0.263 for $K = 22$ with $\delta = 1$ and $\delta = 10^2$, respectively. This is attributed to the method's ability to increase recall while maintaining a competitive precision. Notably, OCCMM achieves a recall of 0.639 for $K = 22$ with $\delta = 10^2$ indicating that it can retrieve approximately 64% of relevant items in the recommendation list. The results also reveal OCCMM flexibility in adapting the recommendation list size based on the δ parameter. As δ increases, the Average_Recommendation metric also rises which reflects its ability to adjust the overlap degree and thus increase the number of items recommended to users. This adaptability effectively allows OCCMM to balance precision and recall and offers customizable recommendation lists based on user preference.

Fig. 5 provides insights into the average of users and items within each co-cluster across varying numbers of co-clusters K for three datasets CiteUlike, LastFm, and Movielens obtained using OCCMM with $\delta = 100$. These visualizations reveal distinct patterns in each dataset, which helps understand the optimal number of co-clusters for an effective recommendation. For CiteUlike Dataset which contains 5551 users and 16980 items, Fig. 5(a) shows that obtained results exhibit a large item-to-user ratio per co-cluster at lower values of K. As K increases, this ratio decreases, leading to a more balanced distribution of users and items across co-clusters. According to Table II, partitioning CiteUlike with $K = 22$ achieves a higher FMeasure (0.318) than with $k = 5$, $K = 10$ and $k = 16$. This suggests that $K = 22$ provides an optimal balance between the size of user-item groups and recommendation accuracy. Higher K values help refine co-clusters, focusing more specifically on user-item interactions which can improve recommendation quality by ensuring neither too large nor too small co-clusters. Concerning the LastFm dataset which has 1226 users and 285 items, we observe a different distribution pattern due to the higher density of users compared to items. Based on Table III, the optimal partitioning for LastFm is at $K = 16$, yielding the highest FMeasure (0.401) among the configurations. This suggests that $K = 16$ strikes a good balance, avoiding excessively large or small co-clusters and providing an effective clustering granularity that enhances recommendation accuracy. For the third dataset, Movielens, which contains 671 users and 9066 items, Fig. 5(c) shows a large item-to-user ratio per co-cluster, similar to CiteUlike. Although the highest FMeasure (0.263) is achieved with $K = 22$, as shown in Table IV, a more suitable choice is $K = 5$, where co-clusters provide a balance of neither too dense nor sparse user-item interactions. $K = 5$ ensures sufficient group density while maintaining recommendation quality, avoiding the sparsity that can occur with higher K values.

VII. CONCLUSIONS AND FUTURE WORKS

We proposed in this paper a novel recommender system framework called Overlapping Co-Clustering with Modularity Maximization, designed to address three persistent challenges in clustering-based recommendations: capturing multifaceted user preferences, enhancing interpretability, and adapting recommendation list sizes to individual contexts. The OCCMM

TABLE IV. EVALUATION METRICS RESULTS OF BPR VS MMFCC VS COCLUS VS OCCMM, MOVIELENS DATASET

Datasets	Algorithms	δ value	Precision	Recall	F_Measure	Avg_Recommendation
$K = 5$	BPR		0.041	0.271	0.070	0.110
	MMFCC		0.107	0.506	0.176	0.125
	COCLUS		0.102	0.503	0.169	0.143
	OCCMM	1	0.110	0.492	0.179	0.078
		10^2	0.106	0.566	**0.178**	0.168
		10^6	0.106	0.566	**0.178**	0.169
$K = 10$	BPR		0.139	0.244	0.177	0.175
	MMFCC		0.098	0.485	0.163	0.119
	COCLUS		0.117	0.455	0.186	0.085
	OCCMM	**1**	0.115	0.504	**0.187**	0.084
		10^2	0.112	0.579	**0.187**	0.179
		10^6	0.112	0.581	**0.187**	0.180
$K = 16$	BPR		0.013	0.231	0.024	0.175
	MMFCC		0.192	0.456	**0.304**	0.103
	COCLUS		0.194	0.410	0.263	0.067
	OCCMM	1	0.173	0.506	0.257	0.088
		10^2	0.156	0.598	0.247	0.178
		10^6	0.150	0.598	0.239	0.180
$K = 22$	BPR		0.013	0.231	0.024	0.175
	MMFCC		0.170	0.528	0.257	0.130
	COCLUS		0.195	0.401	0.262	0.065
	OCCMM	**1**	0.194	0.562	**0.288**	0.117
		10^2	0.166	0.639	0.263	0.198
		10^6	0.166	0.639	0.263	0.198

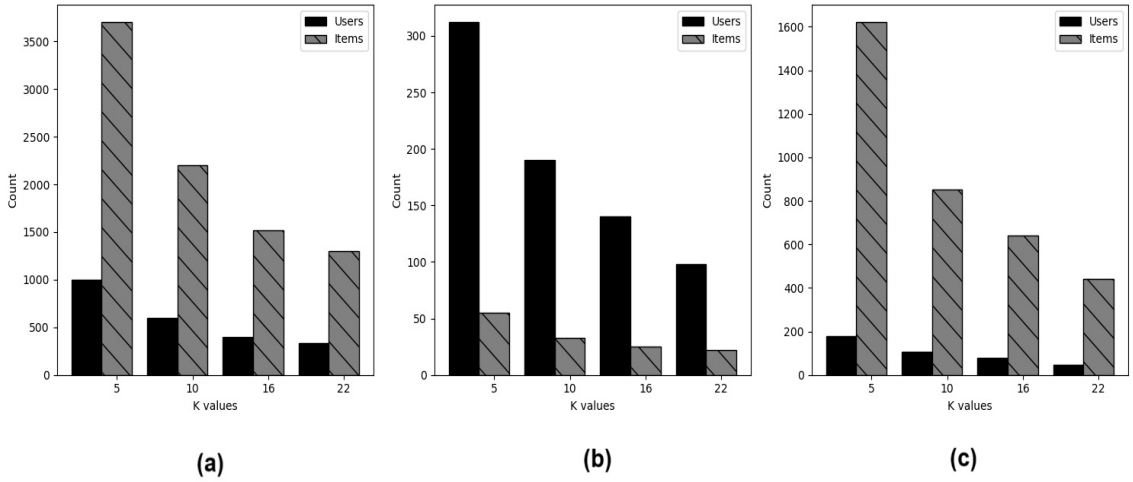

Fig. 5. Average number of Users and Items per Co-cluster across varying K obtained with OCCMM with $\delta = 100$ on: (a) CiteUlike dataset (b) LastFm dataset (c) Movielens dataset.

approach operates in three key steps: 1) implicit data preprocessing, 2) construction of overlapping co-clusters through modularity maximization, and 3) recommendation generation based on these co-clusters. This design effectively balances accuracy, interpretability, and adaptability. By modeling overlapping user–item relationships, OCCMM allows for capturing complex and realistic preference structures. Additionally, it provides transparent rationales for recommendations by using co-clusters rather than simple clusters and also ensures responsiveness to diverse user needs through dynamically adjusting list sizes. Experimental results on real-world datasets confirmed that OCCMM outperforms existing methods in terms of both accuracy and personalization.

While OCCMM demonstrates strong empirical performance, several limitations should be acknowledged. First, the current framework does not explicitly integrate continuous user feedback (implicit or explicit), which may limit its ability to iteratively refine recommendations. Second, the scalability of OCCMM under very large-scale datasets remains an open challenge. Computational complexity could be mitigated through advanced optimization or parallelization strategies. Third, the model is currently evaluated in static environments, whereas many real-world systems require realtime adaptation to dynamic user behaviors. Finally, although OCCMM balances

interpretability and accuracy, its explainability mechanisms are limited to cluster-level reasoning, which may not always capture fine-grained user justifications.

The effectiveness of OCCMM can be improved in future works by investigating other techniques to address critical issues related to recommendation systems. First, the incorporation of explicit and implicit user feedback into the co-clustering process may enable the system to refine recommendations based on user interactions. In addition, optimization techniques can be investigated to improve the scalability of OCCMM when handling large-scale datasets. One can also investigate the integration of OCCMM with other recommendation techniques, such as deep learning or collaborative filtering, to further improve accuracy and robustness. Furthermore, a promising direction to improve the accuracy and robustness of OCCMM is to integrate deep learning-based clustering which can give more accurate results. One can investigate the implementation of real-time recommendation mechanisms within OCCMM to ensure instant adaptation to evolving user behaviors and preferences in live environments.

ACKNOWLEDGMENT

This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. ($UJ - 22 - DR - 48$). The authors, therefore, acknowledge with thanks the University of Jeddah for its technical and financial support.

REFERENCES

[1] J. Das, P. Mukherjee, S. Majumder, and P. Gupta, "Clustering-based recommender system using principles of voting theory," in *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2014, pp. 230–235.

[2] M. Rezghi and E. Baratnezhad, "A novel fuzzy co-clustering method for recommender systems via inverse stereographic nmf," *Expert Systems with Applications*, vol. 259, p. 125301, 2025.

[3] T. George and S. Merugu, "A scalable collaborative filtering framework based on coclustering," in *Fifth IEEE International Conference on Data Mining (ICDM 05)*. IEEE, 2005, pp. 4–11.

[4] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," *SIGMOD Rec.*, vol. 28, no. 2, p. 61–72, Jun. 1999.

[5] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2015, pp. 1235–1244.

[6] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *information retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[7] J. Son and S. B. Kim, "Content-based filtering for recommendation systems using multiattribute networks," *Expert Systems with Applications*, vol. 89, pp. 404–412, 2017.

[8] J. Wu, "Knowledge-based recommender systems: An overview," Oct 2019, last checked on Aug 22, 2020.

[9] Y. Pérez-Almaguer, R. Yera, A. A. Alzahrani, and L. Martínez, "Content-based group recommender systems: A general taxonomy and further improvements," *Expert Systems with Applications*, vol. 184, p. 115444, 2021.

[10] S. Cleger-Tamayo, J. M. Fernández-Luna, J. F. Huete, R. Pérez-Vázquez, and J. C. Rodríguez Cano, "A proposal for news recommendation based on clustering techniques," in *Trends in Applied Intelligent Systems*. Springer, 2010, pp. 478–487.

[11] D. Z. Ulian, J. L. Becker, C. B. Marcolin, and E. Scornavacca, "Exploring the effects of different clustering methods on a news recommender system," *Expert Systems with Applications*, vol. 183, p. 115341, 2021.

[12] N. S. Diana Sousa and L. Spagnolo, "Recommending news articles for public health intelligence," in *The 6th Workshop on Health Recommender System in the 18 th ACM Conference on Recommender Systems (HealthRecSys'24)*, Bari, Italy. ACM, 2024.

[13] N. P. Chadalavada and N. Ramachandran, "An efficient neighbor nodes trust tagging model for optimized route detection in smart grids with secure data transmission," *Heliyon*, vol. 10, no. 11, p. e32074, 2024.

[14] R. Ahuja, A. Solanki, and A. Nayyar, "Movie recommender system using k-means clustering and k-nearest neighbor," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2019, pp. 263–268.

[15] F. Li and C. Zhang, "A deep learning-based recommender model for tourism routes by multimodal fusion of semantic analysis and image comprehension," *Journal of Circuits, Systems and Computers*, 2024.

[16] R. Cao, Z. Li, P. Wei, Y. Tian, and C. Zheng, "A hybrid tourism recommendation system based on multi-objective evolutionary algorithm and re-ranking," in *Advanced Intelligent Computing Technology and Applications*. Singapore: Springer Nature Singapore, 2023, pp. 363–372.

[17] S. Wang, R. Cao, Y. Tian, and C. Zheng, "Hybrid tourism recommendation system: A multi-objective perspective," in *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEExplore, 2022, pp. 1–8.

[18] E. Yu, Z. Ye, Z. Zhang, L. Qian, and M. Xie, "A federated recommendation algorithm based on user clustering and meta-learning," *Applied Soft Computing*, vol. 158, p. 111483, 2024.

[19] M. A. Hosen, S. H. Moz, S. S. Kabir, D. S. M. Galib, and D. M. N. Adnan, "Enhancing thyroid patient dietary management with an optimized recommender system based on pso and k-means," *Procedia Computer Science*, vol. 230, pp. 688–697, 2023, 3rd International Conference on Evolutionary Computing and Mobile Sustainable Networks (ICECMSN 2023).

[20] A. M. A. Al-Sabaawi, H. Karacan, and Y. E. Yenice, "A novel overlapping method to alleviate the cold-start problem in recommendation systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 31, no. 09, pp. 1277–1297, 2021.

[21] W. Zhang and Z. Wu, "E-commerce recommender system based on improved k-means commodity information management model," *Heliyon*, vol. 10, no. 9, p. e29045, 2024.

[22] S. Airen and J. Agrawal, "Movie recommender system using parameter tuning of user and movie neighbourhood via co-clustering," *Procedia Computer Science*, vol. 218, pp. 1176–1183, 2023, international Conference on Machine Learning and Data Engineering.

[23] Z. Wang, J. Chen, J. Li, and Z. Wang, "Interest community-based recommendation via cognitive similarity and adaptive evolutionary clustering," *Chaos, Solitons and Fractals*, vol. 185, p. 115085, 2024.

[24] A. Iftikhar, M. A. Ghazanfar, M. Ayub, S. Ali Alahmari, N. Qazi, and J. Wall, "A reinforcement learning recommender system using bi-clustering and markov decision process," *Expert Systems with Applications*, vol. 237, p. 121541, 2024.

[25] S. Forouzandeh, K. Berahmand, R. Sheikhpour, and Y. Li, "A new method for recommendation based on embedding spectral clustering in heterogeneous networks (reschet)," *Expert Systems with Applications*, vol. 231, p. 120699, 2023.

[26] R. D. and D. M., "A systematic review and research perspective on recommender systems." *J Big Data*, vol. 9, 2022.

[27] R. Heckel, M. Vlachos, T. Parnell, and C. Dünner, "Scalable and interpretable product recommendations via overlapping co-clustering," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 1033–1044.

[28] M. Vlachos, C. Mendler-Dünner, R. Heckel, V. Vassiliadis, T. Parnell, and K. Atasu, "Addressing interpretability and cold-start in matrix factorization for recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 05 2018.

[29] Y. Deldjoo, V. W. Anelli, H. Zamani, A. Bellogín, and T. D. Noia, "A flexible framework for evaluating user and item fairness in recommender systems," *User Model. User Adapt. Interact.*, vol. 31, no. 3, pp. 457–511, 2021.

[30] C. E. Berbague, N. E. islem Karabadji, H. Seridi, P. Symeonidis, Y. Manolopoulos, and W. Dhifli, "An overlapping clustering approach for precision, diversity and novelty-aware recommendations," *Expert Systems with Applications*, vol. 177, p. 114917, 2021.

[31] W. Pan and L. Chen, "Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering," in *Twenty-Third International Joint Conference on Artificial Intelligence*. IJCAI, 2013.

[32] V. Sindhwani, S. Bucak, J. Hu, and A. Mojsilovic, "A family of non-negative matrix factorizations for one-class collaborative filtering problems," in *Proceedings of the ACM Recommender Systems Conference, RecSys*. ACM, 2009.

[33] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," in *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, vol. 23. ACM, 2011, p. 53.