

Complex-Network Theoretic Clustering for Identifying Groups of Similar Listeners in P2P Systems

Amelie Anglade
Centre for Digital Music
Queen Mary, University of London
Mile End Road
London E1 4NS, UK
amelie.anglade@elec.qmul.ac.uk

Marco Tiemann, Fabio Vignoli
Philips Research Europe
High Tech Campus 34
5656 AE Eindhoven, The Netherlands
marco.tiemann@philips.com,
fabio.vignoli@philips.com

ABSTRACT

This article presents an approach to automatically create virtual communities of users with similar music preferences in a distributed system. Our goal is to create personalized music channels for these communities using the content shared by its members in peer-to-peer networks for each community. To extract these communities a complex network theoretic approach is chosen. A fully connected graph of users is created using epidemic protocols. We show that the created graph sufficiently converges to a graph created with a centralized algorithm after a small number of protocol iterations. To find suitable techniques for creating user communities, we analyze graphs created from real-world recommender datasets and identify specific properties of these datasets. Based on these properties, different graph-based community-extraction techniques are chosen and evaluated. We select a technique that exploits identified properties to create clusters of music listeners. The suitability of this technique is validated using a music dataset and two large movie datasets. On a graph of 6,040 peers, the selected technique assigns at least 85% of the peers to optimal communities, and obtains a mean classification error of less than 0.05% over the remaining peers that are not assigned to the best community.

Categories and Subject Descriptors

H.4.3 [Systems and Software]: Distributed systems; I.5.3 [Clustering]: Algorithms; G.2.2 [Graph Theory]: Graph Algorithms

General Terms

Algorithms, Performance

The presented work was performed during an internship of Amelie Anglade at Philips Research Europe, Eindhoven, The Netherlands.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RecSys'07, October 19–20, 2007, Minneapolis, Minnesota, USA.
Copyright 2007 ACM 978-1-59593-730-8/07/0010 ...\$5.00.

1. INTRODUCTION

Music recommender systems and algorithms continue to attract scientific and commercial interest. Generally, music recommender research focuses on either service-based recommender systems, that can be used over the Internet, or on autonomous recommender systems for stand-alone devices such as portable music players. In this paper we are concerned with providing music recommendations and personalized music radio channels directly between connected systems and devices using peer-to-peer networks. Such concepts of peer-to-peer radio emerged recently, and some systems are already publicly available and prove the concept's technical feasibility (e.g. [11]). Regarding personalized recommendation over peer-to-peer networks, relatively little research has been conducted to date. One well-known research project on distributed recommendation is the TRIBLER project (cf. Wang et al. [14]). TRIBLER provides users with personalized recommendations for items shared in peer-to-peer networks using a distributed version of the widely-used collaborative filtering approach (Resnick et al. provide an introduction to collaborative filtering in [13]).

In this paper we describe our approach to identifying peers that share similar music preferences and potentially other criteria and to cluster these peers into communities. Such communities can be used to provide music recommendations and shared music radio channels to peers in these groups. Music shared by peers in such channels can be transmitted directly between peers within the groups or broadcast to the groups of peers depending on the underlying infrastructure.

The paper is organized as follows: In Section 2, the specific problems that need to be addressed to realize distributed clustering of peers by music preferences are discussed. In Section 3, epidemic protocols for identifying similar users in peer-to-peer networks are introduced and evaluated. In Section 4, we perform an analysis of typical datasets for recommender systems, and identify properties of those datasets which can be exploited with graph-based clustering techniques. In Section 5, we comparatively evaluate several such clustering techniques and identify the most suitable according to its performance on optimal assignment of peers to communities. We close with final remarks and pointers to future work in Section 6.

2. PROBLEM DEFINITION AND SOLUTION APPROACH

The main challenge when automatically creating groups or communities of users in the desired setting is to identify clusters of similar peers in a peer-to-peer network so that every peer in the system can enjoy a selection of music that suits her music preferences.

2.1 A dynamical, distributed optimization problem

Creating communities of peers can be expressed as an optimization problem which is composed of several potentially complementary, but also potentially competing objectives described in the following section.

- *Each peer must be satisfied with the music selection proposed to her.* Recommended or played music must match the peer’s personal taste.
- *Within each community a good average satisfaction must be obtained.* Optimally, each peer should be more satisfied with the recommendations provided by her community than she would be in any other community.
- *Adding a new peer to a community must not deteriorate the average satisfaction.* When adding a new peer to a community, the previously stated objective must still be fulfilled both for her and for all members of the community.
- *Extreme community sizes must be avoided.* Both creating very large communities and very small communities must be avoided. Assigning most or all peers to a single large community can result in an unsatisfactory listening experience; fragmenting peers into many very small communities limits the exposure to additional music not included in a peer’s music collection. The community is a way to expose the users to new interesting music matching their music tastes (serendipity).

Moreover, various dynamical aspects have to be taken into account for resolving the optimization problem:

- *Peer-to-peer systems are intrinsically dynamic.* Peers may enter or leave the system at any given time. This may require the system to dynamically adapt the community structure.
- *Music preferences evolve over time.* A peer may temporarily or permanently change her music preferences. Reflecting such changes in a timely manner is important in order to satisfy listeners.

Finally, given the distributed nature of peer-to-peer systems, all computations for creating and maintaining communities must be performed by the peers themselves.

2.2 Solution approach

We adopt a graph-theoretic approach to solve the given problems. We start by distributedly constructing a graph of all peers and then apply a clustering technique on that graph in order to form communities of peers. For constructing this graph, we link each peer to her N most similar peers

in terms of music preferences. Each peer is characterized by a vector of preference-song pairs, where the preference values can be regular (e.g. on a scale from 1 to 5) or binary values (like or dislike). In our case, we assume that these preferences reflect explicit ratings provided by each peer.

We use the Pearson correlation coefficient to define the similarity measure s between two peers a and b . This is a similarity measure that is frequently used in collaborative filtering recommender systems (e.g. Resnick et al. [13]):

$$s(a, b) = \frac{\sum_j (V_{a,j} - \bar{v}_a)(V_{b,j} - \bar{v}_b)}{\sqrt{\sum_j (V_{a,j} - \bar{v}_a)^2 \sum_j (V_{b,j} - \bar{v}_b)^2}} \quad (1)$$

where j is the index of the song for songs present in both peer’s preference vectors, $v_{a,j}$ is the preference value by peer a on song j , \bar{v}_a corresponds to the mean preference value for peer a over her complete song preference vector P_a :

$$\bar{v}_a = \frac{1}{|P_a|} \sum_{i \in P_a} v_{a,i} \quad (2)$$

Using this similarity measure we use a four step approach to find a suitable graph-based solution to automatically and distributedly create communities of peers:

- We implement and compare 3 distributed algorithms to build overlay networks of peers.
- We create and analyze graphs of peers using real-world reference datasets and identify relevant common properties of such datasets.
- We comparatively evaluate several candidate clustering techniques. We then select the most appropriate technique for extracting communities of peers, based on our objectives formulated in Section 2.
- We validate the performance of the selected technique experimentally.

We present selected results of our work in the following sections. A detailed presentation of the work carried out can be found in [2].

2.3 Datasets

Three datasets from the music and movie domain collected under real-world conditions have been analyzed. The datasets vary in the number of users and in the amount of preference data available for them:

1. The *EasyAccess* (EA) dataset gathered within Philips Research contains 74,631 ratings over 5,234 songs from 462 users.
2. The *MovieLens1* (ML1) and *MovieLens2* (ML2) datasets [9] contain respectively 1,000,029 ratings over 3,593 movies provided by 6,040 users and 100,000 ratings of 1,682 movies from 943 users. These datasets are widely used as reference datasets for evaluating recommender algorithms.

All datasets contain regular ratings on a five point numerical scale from one to five. We report evaluation results both for these regular ratings and for binary ratings. Binary ratings are constructed as follows: values larger than 3

are considered as positive ratings, others are considered as negative ratings.

The parameter N for the number of most similar peers used is a variable in the analysis in order to study its influence on the topology of the resulting graphs. In all evaluations we report results for values of $N = 5, 10, 20, 30,$ and 46 (which represents 10% of the population of the smallest dataset). Larger values of N are impractical for distributed solutions: it would be prohibitively time-consuming to compute large numbers of most similar peers over a peer-to-peer system.

3. DISTRIBUTEDLY BUILDING A CONNECTED GRAPH OF USERS

As described in Section 2, our goal is to construct and maintain a constantly evolving graph in which each user of the system is linked to her N most similar peers based on music preferences. For this we need to find the top- N most similar peers for each user, which involves computing $(n(n-1))/2$ similarity measures where n , the number of users in the system, can potentially be quite large (in the order of millions). We need to be able to update these lists of similar peers quickly to reflect new or changed user preferences, which means recomputing the created lists of similar peers every time a user’s preferences are modified.

In a distributed system, peers only have a local view of the system. Thus it is not possible to directly retrieve preference data of the required $n-1$ peers in a distributed system. Epidemic protocol algorithms can be used to spread the required data throughout distributed environments. In this section, we examine several such algorithms to find out whether or not they are suitable for quickly identifying the top- N most similar peers in a peer-to-peer system.

3.1 Distributedly creating a graph

Epidemic protocol algorithms apply mechanisms of epidemic spreads in nature. Having only a local view of a system (for example knowing some other peers), each peer can transmit the pieces of information she has to some other peers through direct local interaction with them, in this way changing their states. We implemented and tested three epidemic protocol algorithms. In the remainder these are referred to as “Buddycast algorithm”, “Modified Buddycast algorithm” (MBuddycast) and “Newscast algorithm” (c.f. [10] for the Newscast algorithm and [14] for the Buddycast algorithm). Each of these algorithms performs three basic tasks for propagating information over the distributed system: maintaining local information, selecting peers to exchange information, and exchanging information.

3.1.1 Maintaining local information

In all three algorithms, each peer maintains a top- N list of peers she considers as the most similar to her among the peers she has encountered in the system, which is referred to as buddy list. It contains the preference lists, addresses and corresponding similarity of the peers (relative to their current preference list). The combination of one peer’s preference list of songs (as defined in Section 2) and her buddy list constitutes her local information that can be updated and transmitted to the other peers. In addition, each peer maintains a cache of a number of addresses of random peers she can contact.

In the “Buddycast algorithm” and “Modified Buddycast algorithm”, each peer also maintains a list of the K most recently visited random peers which are not in the buddy list. This “taboo list” is maintained to ensure that recently visited random peers are not visited again.

3.1.2 Selecting peers to exchange information

The method of selecting peers to exchange information with differs between the three algorithms.

In the “Newscast algorithm”, when a peer p wants to contact another peer, she randomly selects one in the joined list composed of the peers in p ’s buddy list and the peers in p ’s random cache. At each selection step, each peer in this joined list has the same probability to be chosen.

In the “Buddycast algorithm” [14], the buddy list stores the most similar peers as a ranked list. Prior to the selection, the list is appended with N peers from the random cache (excluding the K recently visited random peers from the “taboo list”). A roulette-wheel selection proportional to the rank is applied on this list. This means that higher-ranking peers are more likely selected than lower-ranking peers. The parameter α is a real non-negative number. The parameter value is used to tune the exploitation-to-exploration ratio: the higher the value of α , the more random peers are appended to the list.

In the “Modified Buddycast algorithm”, before selecting a peer to contact, a peer creates an ‘extended buddy list’ as in the Buddycast algorithm. Unlike in the “Buddycast algorithm”, no rank information is used for selecting a peer to contact. Instead a peer is selected randomly from the ‘extended buddy list’.

3.1.3 Exchanging information

When one peer contacts another one, she obtains information from her and updates her own information accordingly. In the three epidemic protocol algorithms described here if a peer p_1 selects a peer p_2 to communicate with, p_2 sends p_1 her buddy list as well as her own preference list. p_1 evaluates her similarity with p_2 ’s buddies and p_2 using their preference lists. From this joined list of up to $2N+1$ buddies she retains the N most similar to her as her new buddy list.

If p_2 is a random peer, p_1 ’s list of the K most recently visited peers is updated to include p_2 as the most recently visited random peer. p_1 ’s random cache (of size C) is updated as well, so that it contains C peers that are not in p_1 ’s new buddy list.

3.2 Algorithm performance comparison

An epidemic protocol is suitable for distributed recommendation if the list obtained by the peers using the epidemic protocol (*epidemic information*) tends to converge with the information of a centralized algorithm (*exact information*) quickly. In simulations, the number of protocol iterations can be used to measure the time required to achieve this. One iteration of an epidemic protocol denotes updating the buddy list of every peer in the system exactly once. For a performance comparison, each buddy list obtained with the epidemic algorithms is compared to the exact buddy list of the same peer obtained with a centralized algorithm after each iteration of the epidemic algorithm. The fraction of overlap f_0 at this iteration is the number of exact bud-

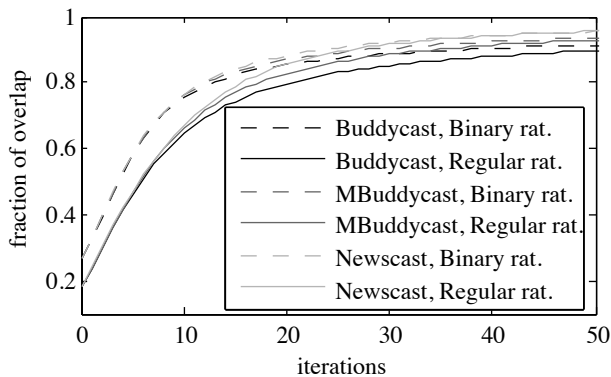


Figure 1: Convergence of the three epidemic protocols to the results obtained with a centralized algorithm applied on EA for both kinds of ratings, $N = 10$, $K = 10$, $\alpha = 1$. Iterations 0 to 50.

dies found by the epidemic algorithm divided by the total number of exact buddies.

$$f_0 = \frac{\sum_{p \in P} |\text{Epidemic buddies}(p) \cap \text{Exact buddies}(p)|}{\sum_{p \in P} |\text{Exact buddies}(p)|} \quad (3)$$

where P is the ensemble of all peers.

The simulations were performed using the EasyAccess dataset (described in Section 2.3). Parameter values for the epidemic algorithms were chosen equal to the values used in [14]: $N = 10$, $K = 10$ and $\alpha = 1$ (note that K and α are not used in the Newscast algorithm). No dynamics were introduced in these simulations: all peers were accessible during the entire simulation, and all peers' preference values remained unchanged.

The results obtained in these simulations are shown in Figure 1 and Figure 2. Each graph represents the average fraction of overlap over 30 trials for each epidemic protocol. All three epidemic algorithms converge to the results of the centralized algorithm very well. At least 90% of the exact buddies are found in less than 50 iterations, which is a sufficiently small number of iterations. For larger numbers of iterations, all tested epidemic protocols reach very high values of overlap between 98.22% for the least effective protocol and up to 99.08% for the best protocol with 500 iterations.

When comparing results for binary versus regular ratings, for small numbers of iterations epidemic algorithms using binary ratings outperform the ones using regular ratings. For larger numbers of iterations however, the epidemic algorithms obtain similar results with both types of ratings. The epidemic protocol algorithm experiments with regular ratings perform equal or better than in experiments with binary ratings after 45 to 250 iterations varying by the algorithm used. The choice of using regular or binary ratings thus mainly depends on the epidemic algorithm chosen and on the number of iterations that can be performed before giving recommendations.

Finally, the simulations show that the Newscast algorithm obtains more accurate results than the Modified Buddycast algorithm which in turn outperforms the Buddycast algorithm. This is surprising given the results obtained in [14], where significantly better results are obtained with the Bud-

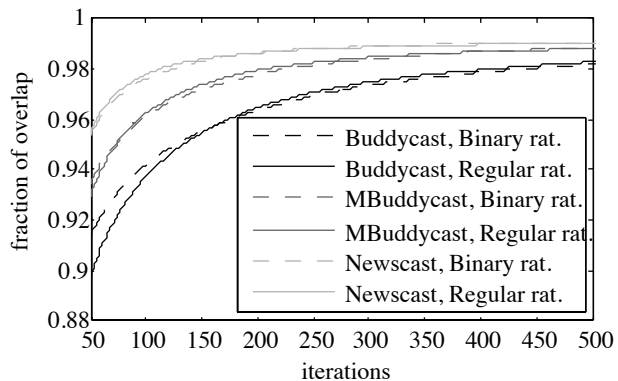


Figure 2: Convergence of the three epidemic protocols. Iterations 50 to 500.

dycast algorithm than with the Newscast algorithm (using implicit binary ratings) on a different dataset containing 480 peers. Additional simulations using datasets with varying properties need to be performed to determine possible causes of these differences in results. However, as already pointed out, all three algorithms perform sufficient and quite well on our test dataset. For this dataset, either of the three is suited to construct a graph of similar users efficiently.

To verify whether these epidemic protocols would perform equally well on peer-to-peer networks of larger sizes and for a larger number of buddies per peer, a simulation using the Modified Buddycast algorithm was carried out using the MovieLens1 dataset (see Section 2.3). For this simulation, regular ratings were used, and the parameters $N = 30$, $K = 30$ and $\alpha = 1$ were chosen. The evolution of the average fraction of overlap of this simulation is shown in Figure 3. The results are encouraging: after 50 iterations, the Modified Buddycast algorithm is able to correctly identify 85% of the 30 6040 top-N peers of the network. The convergence of the epidemic protocol to the exact solution here is almost as rapid as the convergence of the same epidemic protocol on a much smaller dataset with three times less peers per user to examine. Thus, the three implemented epidemic protocols can be considered to be scalable up to very large graphs of peers with little degradation in performance.

4. GRAPH OF PEERS

4.1 Graph properties

Peer-to-peer systems can easily contain thousands or millions of peers. The recently established scientific field of complex network theory is concerned with studying and simulating complex relations and dynamics in large real-world networks [15]. Complex network theory is based on graph-theoretic foundations and is specifically concerned with real-world phenomena that exhibit network properties [1]. Thus insights and solutions from this field may prove helpful for addressing the problems we are interested in.

A graph can be constructed as a *directed graph*, in which all edges between nodes are directed, or as an *undirected graph*, in which edges are not directed. A *component* is a maximal sub-graph of nodes for which a path exists between

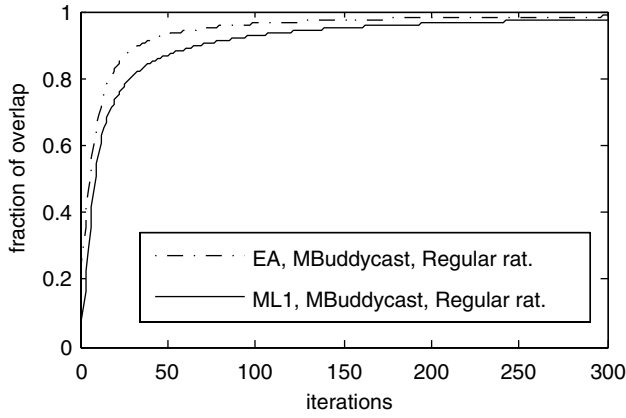


Figure 3: Evolution of the fraction of overlap of the Modified Buddycast algorithm applied to ML1 (for $N = 30$, $K = 30$ and $\alpha = 1$) and EA (for $N = 10$, $K = 10$ and $\alpha = 1$) datasets. Regular ratings, iterations 0 to 300.

every pair of nodes. A *connected graph* is a graph that has one unique component: a path exists for every pair of nodes of this graph. A *disconnected graph* consists of several separated components. The *distance* d_{ij} between two nodes i and j is the length (i.e. the number of edges) of the shortest path between the two nodes. The distance between nodes in two components is infinite. The *average distance* l in a graph is the mean distance between node pairs in the graph. The *neighborhood* of a node i are the nodes immediately connected to it and its *degree* k_i is the number of edges connected to it. In a directed graph the *indegree* and *outdegree* are the number of incoming and outgoing edges of a node.

4.2 Graph components and average distance

The “most similar peers”-relation used to construct graphs of peers is not a symmetric relation. It is most suitably represented as a directed graph of peers. However, directed graphs created from the evaluation datasets are disconnected graphs for all experimental conditions. The resulting graphs consist of one giant component and numerous small components of size at most of the order of $\ln(n)$, where n is the number of users (cf. Table 1). This behavior is commonly found in real-world networks and has been studied in [6].

Given the objectives defined in Section 2, a large fraction of the small components found must be considered as too small to constitute communities by themselves. Therefore we use undirected graphs by discarding all information about directedness of edges. This transformation is commonly used in complex networks analysis. It can be applied when it is reasonable to assume that connections expressed by edges can be considered to be symmetric, as it is the case for user similarities using item ratings (even if it adds for some nodes some neighbors which are not part of their nearest neighbors). The resulting undirected graphs are fully connected for all experimental conditions. The average distance between nodes in the resulting undirected graph is no longer infinite but in the range between 2 and 3 (cf. Table 1). Notice that the transformation from directed to undirected graphs did not have any noteworthy effect on the other properties of the graphs (i.e. clustering coefficient and degree distribution).

	$s\mathcal{C}$	k	l_r	l	\mathcal{C}_r	\mathcal{C}
EA $N = 5$	4	9.5	2.73	2.64	0.0205	0.1041
EA $N = 10$	2	17.8	2.13	2.36	0.0384	0.1041
EA $N = 20$	1	31.9	1.77	2.10	0.0690	0.1012
EA $N = 46$	1	62.7	1.48	1.87	0.1357	0.1659
ML2 $N = 5$	6	9.6	3.02	2.87	0.0102	0.0551
ML2 $N = 10$	6	18.7	2.34	2.55	0.0198	0.0503
ML2 $N = 20$	2	35.3	1.92	2.22	0.0374	0.0590
ML2 $N = 46$	1	73.0	1.60	1.95	0.0774	0.1132
ML1 $N = 5$	3	10.0	3.79		0.0017	0.1200
ML1 $N = 10$	3	19.9	2.91		0.0033	0.1013
ML1 $N = 20$	5	39.4	2.37		0.0065	0.0975
ML1 $N = 46$	3	89.1	1.94		0.0148	0.1002

Table 1: Size $s\mathcal{C}$ of the second biggest component in the directed graph. Average degree k , average distance l and clustering coefficient \mathcal{C} of the undirected graphs of peers for binary ratings and several values of number of peers N . The average distance is missing for the ML1 dataset due to computing complexity. l_r and \mathcal{C}_r are the average distance and the clustering coefficient of a random graph of the same size and same average degree.

4.3 Clustering coefficient

The *clustering coefficient* \mathcal{C}_i of a node i measures the tendency of the neighbors of a node i to be also neighbors of each other. Watts and Strogatz [16] define it as the number of links E_i between the neighbors of node i divided by the total number of links that can possibly exist between those neighbors $(k_i(k_i - 1))/2$. The clustering coefficient \mathcal{C} of a graph is the average of the clustering coefficients of all its nodes. A *clique* is a set of nodes in a graph in which every node is the neighbor of every other node; as a consequence, the clustering coefficient of a clique is 1.

The clustering coefficient measures whether groups (or cliques) of nodes tend to be present in a graph. This is a potentially interesting property given the objectives in Section 2. Frequent occurrences of cliques may be exploited for creating communities of peers. For the evaluation datasets, clustering coefficients are indeed consistently larger, and often much larger, than clustering coefficients of a random graph with the same number of nodes and average degree (cf. Table 1). This observation is independent of the size of the dataset analyzed. Peers are highly clustered irrespectively of the size of the network. However, undirected graphs of most similar peers also exhibit small average distances. This combination of properties indicates that the analyzed graphs have properties of small-world networks. Small-world networks are frequently encountered when analyzing real-world phenomena [16]. They tend to form a single cluster with very small average distances. Here this cluster tends to be very compact since the connection density (linked to N) is high. This makes it difficult to apply traditional clustering techniques successfully.

4.4 Degree distribution

Since the small average distance within the graphs of users makes it difficult to apply clustering techniques effectively, we study the degree distribution of the created graphs to find other potentially useful properties. The undirected graphs

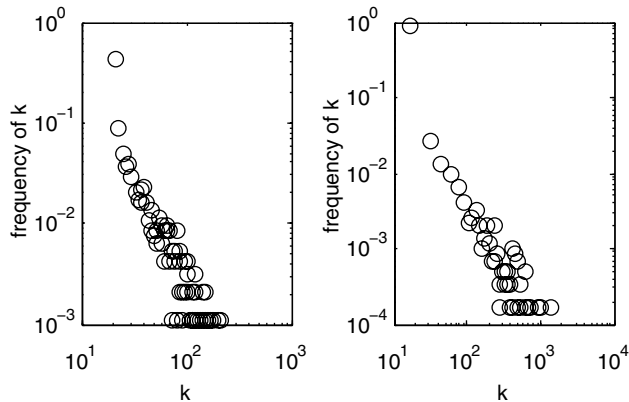


Figure 4: Degree distribution of the MovieLens2 graph for binary ratings and $N = 20$ (left) and MovieLens1 graph for regular ratings and $N = 10$ (right). On a log-log scale it can be approximated by a straight line (power law distribution) or a broken line of negative slope.

consistently display the same characteristics across all simulation conditions, as indicated in Figure 4. While the degree distribution of the graphs is not a power-law distribution, it is similar to one (as shown in Figure 4), and indeed the graphs exhibit properties similar to those of scale-free networks.

A scale-free network is a graph following a power-law distribution defined by $P(k) \sim k^{-\alpha}$ [3]. One property of such networks that can be useful for us is that connections between nodes in scale-free networks are not as evenly distributed as they would be in random graphs. Instead, scale-free networks contain some nodes that are highly connected to many other nodes, termed “hubs”, and many nodes with very small degrees. The graphs obtained from the EA, ML1 and ML2 datasets exhibit exactly such properties. This presence of hubs can be exploited for cluster analysis using specific techniques as discussed in Section 5.

4.5 Robustness to node removal

Another property of scale-free networks is their *resilience*: scale-free networks are robust to random node removal, as long as the removed random node is not a hub node. Hub node removal however can have a large impact on a scale-free network, potentially splitting the network into several components. As already mentioned in Section 4.2, it would be difficult to create and maintain communities in graphs with several components.

To analyze the robustness to node removal, nodes are randomly removed from the graphs for different values of N (see Figure 5). For neighborhood sizes of 10 or larger, the network proves to be highly robust to random node removal: even when 50% of the nodes are removed from the network simultaneously, graphs still consist of one unique component. Smaller neighborhood sizes lead to a higher sensitivity to node removal. Interestingly, the graphs also prove to be relatively robust to hub removal. This shows that while the graphs do contain hub nodes, they only partially exhibit properties of scale-free networks.

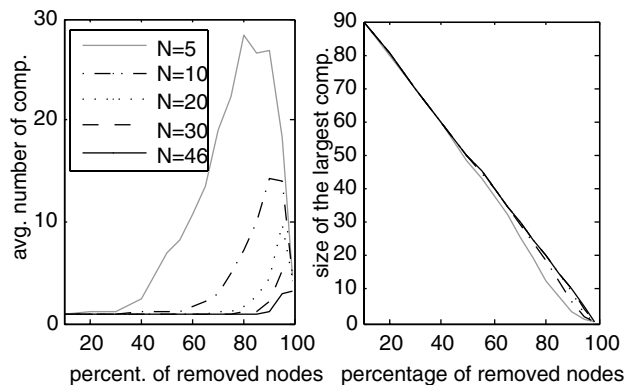


Figure 5: Robustness of the EA graph to node removal. Average number of components and percentage of nodes of the initial graph in the largest component over 50 trials as a function of the percentage of randomly selected removed nodes.

5. A CLUSTERING TECHNIQUE FOR EXTRACTING COMMUNITIES

Given the highly compact nature of the analyzed graphs, selecting a suitable clustering technique to extract communities of peers is not a straightforward task. Well-studied hierarchical clustering techniques, for example, can not be readily applied given this property, because the graphs are too compact to be divided.

5.1 Considered clustering techniques

We evaluated two well-known graph-partitioning clustering algorithms for the task of clustering peers: the Max-Flow technique introduced by Flake et al. [7], and the Edge-Betweenness clustering algorithm introduced by Girvan and Newman [8].

The Max-Flow algorithm is based on a source-sink procedure that starts from a representative set of seed nodes (source), and extracts a cluster by cutting the minimum number of edges linking it to the rest of its graph (sink).

The Edge-Betweenness technique is a hierarchical divisive clustering technique. Edges are progressively removed from a graph to reveal clusters. Assuming that the density of edges is higher within a cluster than between clusters, Girvan and Newman show that all shortest paths between communities go along inter-cluster edges. Based hereon they define edge-betweenness as “the number of shortest paths between a pair of vertices that run along it” [8], and progressively remove the edges with the highest edge-betweenness to reveal clusters.

Besides applying these graph-divisive algorithms we can exploit the presence of hubs in the evaluated graphs to cluster them. They exhibit several interesting properties. Since the outdegree of a node is fixed to N , a hub is a node with a high indegree, meaning that a hub is one of the most similar users for a large number of users. Moreover, the average number of song preferences per hub is always considerably smaller than the average number of songs per user over the whole network, with hub users having around a third of this value. These few songs are also liked by the hubs’ neighbors. The presence of hub users can be explained as an effect of

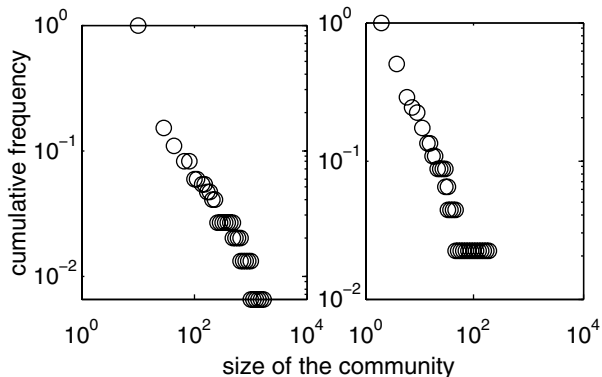


Figure 6: Cumulative distribution of the sizes of the communities for binary ratings, $N = 20$, 150 starting hubs ML dataset (left) and for regular ratings, $N = 46$, 46 starting hubs EA dataset (right) using Hub-Based clustering.

the Pearson correlation, which is computed on present preference data only, ignoring missing values. Thus, a hub user will be a user whose profile contains a small amount of preference data that is a subset of many other users' profiles. So a clustering technique that exploits the idea that hubs can be considered as centers of the communities is also evaluated: the Hub-Based clustering technique introduced by Da Fontoura Costa [5].

In Hub-Based clustering, each hub in a large network is considered as the centroid of a cluster. Starting from these centroids, clusters are obtained by propagating labels from the hubs to other nodes through shortest paths. This procedure is performed as simultaneous wave fronts and repeated until every node is labeled. The desired number of communities N_c is set by choosing the number of starting hubs which is a parameter of the system. The N_c starting hubs are the nodes with the highest degrees in the graph. This technique can easily be applied in distributed environments, since the labels are propagated through purely local interactions between users and their direct neighbors. Experiments with edges that were assigned with correlation values as weights were also performed, but had undesirable effects on the sizes of the created communities.

5.2 Evaluation of clustering techniques

The three clustering techniques are applied on the described datasets, again evaluating all experimental conditions described in Section 2.3.

Evaluation results of both the Max-Flow and the Edge-Betweenness technique are similar across all experimental conditions: some individual nodes and few very small clusters (of 2-3 nodes) are isolated from the rest of the graph, which remains as a single giant component. This is not sufficient given the objectives defined in Section 2.

The evaluation of Hub-Based clustering shows that this technique does not generally lead to such extreme cluster sizes. Figure 6 shows cluster sizes created using Hub-Based clustering for the EasyAccess and MovieLens1 datasets. Hub-Based clustering results in a larger range of different cluster sizes for all experimental conditions. It satisfies the requirement to avoid only creating extreme cluster sizes (see Section 2). Moreover, the cumulative distributions of the clus-

ter sizes are very similar to the degree distributions of the networks of most similar peers. On a log-log scale they can be approximated by a straight line (power law distribution) or a broken line of negative slope. Here the created cluster structures from the Hub-Based clustering technique seem to closely reflect the intrinsic topology of the network of peers.

In conclusion, both the Max-Flow and the Edge-Betweenness technique only create clusters of extreme sizes. This is likely caused by the specific properties found in the evaluation datasets. The compactness of the graphs can not be divided into clusters sufficiently by these two techniques. Hub-Based clustering creates a range of different cluster sizes, which makes it a suitable candidate for further validation.

5.3 Validation measure

Validating the suitability of the Hub-Based clustering technique to create clusters or communities of peers according to our objectives requires us to assess the quality of the clustering achieved.

A frequently used quality measure for clustering techniques is the *modularity* Q introduced by Girvan and Newman [12]. The *modularity* determines whether a cluster structure tends to have dense connections within clusters and sparser connection between them. For the highly compact graphs of the evaluation datasets (see Section 4.3) using Q to evaluate community formation is not feasible. The graphs of the evaluation datasets are too compact for any clustering technique to achieve expected values for good clustering results using Q . The *modularity* Q is not well-suited for this validation, at least when considering standard interpretations of *modularity* values.

Instead, we evaluate the quality of the communities of peers we obtain by measuring the peers' satisfaction with their assignment to a cluster. To evaluate the satisfaction using the evaluation datasets, we proceed as follows. For a community we define the preference list of songs to be a set containing all songs of all members of the cluster. The average rating for a song is used as the community score of a song. For each user u a truncated preference list of her community C_u is computed. This truncated preference list is composed of the ratings of all users of the community except for those of user u . To measure how satisfied a user is by her community, we evaluate whether the song selection of the community she was assigned to without her being part of it suits her better than the music of other communities. A user u is satisfied by her community if:

$$s(u, C_u) \geq \frac{1}{|C| - 1} \sum_{c \in C - \{C_u\}} s(u, c) \quad (4)$$

where C is the ensemble of all communities and s is the Pearson correlation coefficient computed on the preference lists. The classification error e , made when a user is not satisfied by her community (misclassified), is:

$$e = \frac{\mu - s(u, C_u)}{2} \quad (5)$$

where

$$\mu = \frac{1}{|C| - 1} \sum_{c \in C - \{C_u\}} s(u, c) \quad (6)$$

5.4 Validation results

Our validation of the clusters found using the Hub-Based clustering technique results in several noteworthy findings. Firstly, the percentage of well-classified users varies between 48% and 92%. It increases with the size of the network. Using regular ratings leads to better classification results than using binary ratings. The neighborhood size chosen also influences the classification accuracy: the percentage of satisfied users reaches its maximum values for numbers of most similar peers $N = 20$, and $N = 30$. It is consistently above 75% for all datasets used for validation for those values of N when they are combined with regular ratings and an optimal number of starting hubs chosen so that the average number of users per community is between 20 and 30 (cf. [2]). For the largest dataset used this percentage is consistently above 85%. This indicates that the percentage of correctly assigned peers increases with the overall community size.

The classification error for the remaining users varies between 0.03 and 0.16. Again, this error decreases as the size of the dataset increases and is consistently smaller than 0.05 for the best parameters described above. This means that the music of their community is almost as good for them as the music from the other communities.

Summarizing our evaluation, we find that the evaluated Hub-Based clustering technique assigns a large majority of users to the clusters optimal to them and assigns the non-optimally assigned users to clusters that are very close to optimal clusters for neighborhood sizes of 20 and 30 most similar peers. The results of the validation indicate that the technique performs with increasing quality the larger the dataset it is applied on is.

To confirm these encouraging results we plan to compare the users' subjective satisfaction when creating communities using the Hub-Based clustering technique with their satisfaction when applying other well-known clustering techniques such as k-means clustering.

6. CONCLUSION

In this paper we presented a technique to cluster users according to their explicitly or implicitly stated music preferences. In addition to dealing with the highly compact nature of the graphs of similar peers this technique also reflects and uses the intrinsic topology of the created graphs to build communities. It did so with convincing results, consistently assigning at least 85% of all peers to optimal communities, and assigning remaining users to communities that are almost as suitable as their optimal community. The graph of peers created using the Pearson correlation measure proved to be highly robust to random node removal. Graphs with such properties are well suited to be used in peer-to-peer networks. We showed how such graphs can be efficiently and quickly created in peer-to-peer networks with epidemic protocols.

Future work includes investigating the performance of the proposed solution in dynamic environments with rapidly joining and parting network members, and with user preferences that change over time. Another interesting challenge that remains is to develop highly efficient means to identify starting hubs for Hub-Based clustering in a peer-to-peer environment, with which a low-overhead distributed implementation of this clustering technique can be provided. We also intend to use different similarity measures for in-

stance to attenuate any possible popularity bias that might appear with the Pearson correlation coefficient (e.g. by including the inverse user frequency suggested by Breese *et al.* [4]). We plan to examine the influence of these different similarity measures on the graph structures created, and to evaluate their performance and applicability in peer-to-peer networks.

7. ACKNOWLEDGMENTS

The authors acknowledge the support of the UK Engineering and Physical Research Council (EPSRC) for support of the OMRAS2 project (EP/E017614/1) in which Amélie Anglade is currently involved. The authors would like to thank the GroupLens Research Group for providing the MovieLens ratings dataset.

8. REFERENCES

- [1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.
- [2] A. Anglade. Virtual communities for creating shared music channels. Master's thesis, Chalmers University of Technology, Göteborg, Sweden, 2007.
- [3] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- [4] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, 1998.
- [5] L. da F. Costa. Hub-based community finding. *arXiv:cond-mat/0405022*, 2004.
- [6] P. Erdős and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17–61, 1960.
- [7] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3):66–71, 2002.
- [8] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proc. Natl. Acad. Sci. USA*, volume 99, pages 7821–7826, 2002.
- [9] <http://www.grouplens.org/>.
- [10] M. Jelasity and M. van Steen. Large-scale newscast computing on the internet. Technical Report IR-503, Vrije Universiteit, Amsterdam, 2002.
- [11] <http://www.mercora.com/>.
- [12] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 2004.
- [13] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of the Conference on Computer Supported Cooperative Work*, pages 175–186, New York, 1994. ACM.
- [14] J. Wang, J. A. Pouwelse, J. Fokker, and M. Reinders. Personalization of a peer-to-peer television system. In *Proc. of EuroITV 2006*, Athens, 2006.
- [15] D. Watts. *Six Degrees: The Science of a Connected Age*. Norton, 2003.
- [16] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.