

VIRTUAL COMMUNITIES FOR CREATING SHARED MUSIC CHANNELS

Amélie Anglade¹, Marco Tiemann, Fabio Vignoli

Philips Research Europe

High Tech Campus 34

5656 AE Eindhoven, The Netherlands

amelie.anglade@elec.qmul.ac.uk, {marco.tiemann, fabio.vignoli}@philips.com

ABSTRACT

We present an approach to automatically create virtual communities of users with similar music tastes. Our goal is to create personalized music channels for these communities in a distributed way, so that they can for example be used in peer-to-peer networks. To find suitable techniques for creating these communities we analyze graphs created from real-world recommender datasets and identify specific properties of these datasets. Based on these properties we select and evaluate different graph-based community-extraction techniques. We select a technique that exploits identified properties to create clusters of music listeners. We validate the suitability of this technique using a music dataset and a large movie dataset. 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.

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 non network-enabled 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. The concept of peer-to-peer radio emerged recently. Such systems are already publicly available and prove its technical feasibility (e.g. [9]). Regarding personalized recommendation over peer-to-peer networks relatively little research has been conducted to date. One such research project on distributed recommendation is

¹ Now with Queen Mary University of London, Centre for Digital Music, Mile End Road, London E1 4NS, UK; M.Sc. student at Chalmers University of Technology, Department of Applied Physics, Göteborg, Sweden during the study.

the TRIBLER project (cf. Wang et al. [12]). 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 [11]). We describe in this paper our approach to cluster peers into groups that share similar music preferences and potentially other criteria, and to provide music recommendations and shared music radio channels to these groups. Music shared by peers in such channels can then be transmitted directly between peers within the groups or broadcasted to the groups of peers depending on the underlying infrastructure.

The paper is organized as follows: In Section 2, we discuss the specific problems that need to be addressed to realize distributed clustering of peers by music preferences. In Section 3, we perform an analysis of typical datasets for recommender systems, we identify properties of those datasets which can be exploited by using graph-based clustering techniques. In Section 4, we comparatively evaluate several 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 5.

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

2.1 A dynamical, distributed optimization problem

- *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.

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. We provide an in-depth discussion of solutions for this aspect elsewhere [2].

2.2 Solution approach

We adopt a graph-theoretic approach to solve the given problems. We start by 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 numerical (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. [11]):

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

where j is the index of the song for songs present in both peer’s preference vectors, $v_{a,i}$ is the preference value by

peer a on song i , 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 three step approach to find a suitable graph-based solution to automatically create communities 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].

3 GRAPH OF PEERS

3.1 Datasets

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

1. The *EasyAccess* (EA) dataset gathered within Philips Research contains 74,631 ratings provided by 462 users over 5,234 songs from 234 artists. It does not contain any content metadata.
2. The *MovieLens* (ML) dataset [8] contains 1,000,029 ratings over 3,593 movies provided by 6,040 users. The MovieLens dataset is widely used as a reference dataset for evaluating recommender algorithms.

Additional evaluation results for a medium-sized dataset are available in [2].

Both 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 strictly larger than 3 are considered as positive ratings, others are considered as negative ratings. This results in an approximately equal amount of positive and negative ratings without having to discard rating data.

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 . Larger values of N are impractical for distributed solutions: it would be prohibitively time-consuming to compute the exact N most similar peers for each user for large

values of N over a peer-to-peer system (especially for large datasets as the time needed to compute these similar peers increases with the size of the dataset and with N).

3.2 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 [14]. 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 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.

3.3 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 [5].

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. 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).

	sc	$\langle k \rangle$	l_r	l	C_r	C
EA $N=5$	4	9.5	2.73	2.64	0.02	0.10
EA $N=10$	2	17.8	2.13	2.36	0.04	0.10
EA $N=20$	1	31.9	1.77	2.10	0.07	0.10
EA $N=46$	1	62.7	1.48	1.87	0.14	0.17
ML $N=5$	3	10.0	3.79		0.0017	0.12
ML $N=10$	3	19.9	2.91		0.0033	0.10
ML $N=20$	5	39.4	2.37		0.0065	0.10
ML $N=46$	3	89.1	1.94		0.015	0.10

Table 1. Size of the second biggest component sc in the **directed graph**. Average degree $\langle k \rangle$, average distance l and clustering coefficient 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 ML data set due to computing complexity. l_r and C_r are the average distance and the clustering coefficient of a random graph of the same size and same average degree.

3.4 Clustering coefficient

The *clustering coefficient* 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 [13] define it as the number of links E_i between the neighbors of a node divided by the total number of links that can possibly exist between those neighbors $\frac{k_i(k_i-1)}{2}$. The clustering coefficient 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 form 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 [13]. They tend to form a single very compact 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.

3.5 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

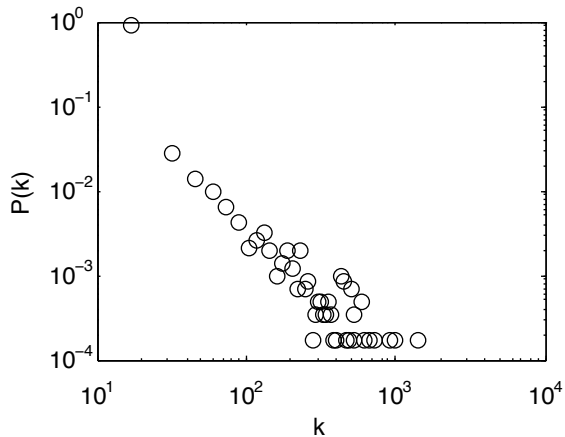


Figure 1. Degree distribution of the MovieLens graphs for **regular ratings** and $N = 10$. On a log-log scale it can be approximated by a straight line (power law distribution) or a broken line of negative slope.

undirected graphs consistently display the same characteristics across all simulation conditions, as indicated in Figure 1. While the degree distribution of the graphs is not a power-law distribution, it is similar to one (as shown in Figure 1), 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 $P(k) \sim k^{-\gamma}$ [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 and ML datasets exhibit exactly such properties. This presence of hubs can be exploited for cluster analysis using specific techniques as discussed in Section 4.

3.6 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 3.3, 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 2). 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.

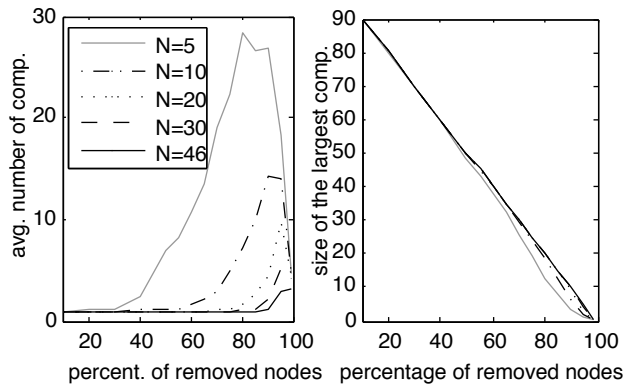


Figure 2. Robustness of the EasyAccess graph to node removal. Average number of components and percentage of nodes of the initial graph in the largest component (i.e. percentage of nodes of the initial graph in the largest component) over 50 trials as a function of the percentage of randomly selected removed nodes.

4 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.

4.1 Considered clustering techniques

Two well-known graph-partitioning clustering algorithms were evaluated for the task of clustering peers: the Max-Flow technique introduced by Flake et al. [6], and the Edge-Betweenness clustering algorithm introduced by Girvan and Newman [7].

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” [7], and we progressively remove the edges with the highest edge-betweenness to reveal clusters.

Besides these graph-divisive algorithms we can try to 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, so a user similar to a high number of other ones. 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,

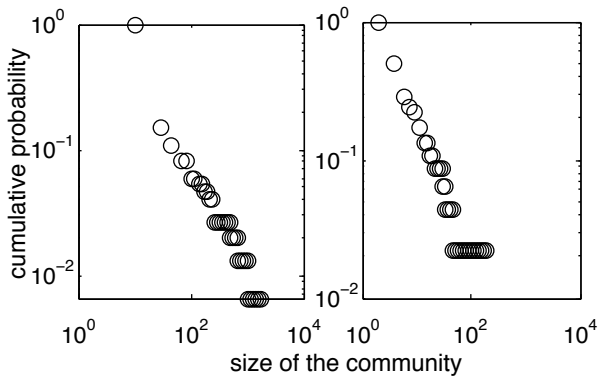


Figure 3. Cumulative distribution of the sizes of the communities for binary ratings, $N = 20$, 150 starting hubs ML data-set (left) and for regular ratings, $N = 46$, 46 starting hubs EA data-set (right) using Hub-Based clustering.

with hub users having around a third of this value. Hubs can be seen as users with songs commonly loved by their neighbors, and so can federate around them communities of users with common tastes. Thus a clustering technique that exploits this idea is also evaluated: the Hub-Based clustering technique introduced by Da Fontoura Costa [4].

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. 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 of this last technique with edges that were assigned with correlation values as weights were also performed, but had undesirable effects on the sizes of the created communities.

4.2 Evaluation of clustering techniques

The three clustering techniques are applied on the described datasets, again evaluating all experimental conditions described in Section 3.1.

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 3 shows cluster sizes created using Hub-Based clustering for the EasyAccess and MovieLens 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 cluster sizes are very similar to the de-

gree 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.

4.3 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 [10]. 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 3.4) 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:

$$s(u, C_u) \geq \frac{1}{|C| - 1} \sum_{c \in C - \{C_u\}} s(u, c) \quad (3)$$

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:

$$e = \frac{\mu - s(u, C_u - \{u\})}{2} \quad (4)$$

where

$$\mu = \frac{1}{|C| - 1} \sum_{c \in C - \{C_u\}} s(u, c) \quad (5)$$

4.4 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.

5 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 it also respects and uses the intrinsic topology of those 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 showed to be highly robust to random node removal. Graphs with such properties are well suited to be used in peer-to-peer networks.

Future work we are planning includes investigating how the proposed solution performs in dynamic environments with many rapidly joining and parting network members, and in which user preferences change over time. We also intend to examine the influence of different similarity measures on the graph structure, and to evaluate the

performance and applicability of different similarity measures in peer-to-peer networks.

6 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 involved. The authors would like to thank the GroupLens Research Group for providing the MovieLens ratings dataset.

7 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] L. da F. Costa. Hub-based community finding. *arXiv:cond-mat/0405022*, 2004.
- [5] P. Erdős and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17–61, 1960.
- [6] 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.
- [7] 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.
- [8] <http://www.grouplens.org/>.
- [9] <http://www.mercola.com/>.
- [10] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 2004.
- [11] 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.
- [12] J. Wang, J. A. Pouwelse, J. Fokker, and M.J.T. Reinders. Personalization of a peer-to-peer television system. In *Proc. of EuroITV 2006*, Athens, 2006.
- [13] D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, 1998.
- [14] D.J. Watts. *Six Degrees: The Science of a Connected Age*. Norton, 2003.