Guiding Users within Trust Networks Using Swarm Algorithms

Mihaela Breaban, Lenuta Alboaie, Henri Luchian
"Alexandru Ioan Cuza" University
Faculty of Computer Science Iasi, Romania
{pmihaela,adria,hluchian}@infoiasi.ro

Abstract—This paper is concerned with a problem in information organization and retrieval within Web communities. Most work in this domain is focused on reputation-based systems which exploit the experience gathered by previous users in order to evaluate resources at the community level. The current research focuses on a slightly different approach: a personalized evaluation system whose goal is to build a flexible and easy way to manage resources in a personalized manner. The functionality of such a model comes from local trust metrics which propagate the trust to a limited level into the system and, finally, lead to the appearance of minorities sharing some similar features/preferences. A modified PSO procedure is designed in order to analyze such a system and, in conjunction with a simple agglomerative clustering algorithm, identify homogenous groups of users.

I. INTRODUCTION

The World Wide Web stores great amounts continuously increasing data. Any kind of resources can be published by anyone: a diary published within a blog; a track that a user wants to share with the others; a study that the author wants to make public. In this context, the main problem is not the lack of good quality resources, but their retrieval, organization and maintenance. This study is part of a recently proposed approach to improve the performance of existing management systems within online communities. An online community can be a forum, a group of blogs or other social system which allows users to interact and to share resources.

The problem we tackle is related to the Web Collective Intelligence spectrum, where many study directions can be envisaged. Much work is carried out nowadays in order to design strategies to rate the resources or to build recommendation systems.

One example is Google Page Rank (A. Langville, C. Meyer, 2006) which, based on some complex algorithms, assigns ranks to Web pages as a measure of their "global importance".

In the case of recommendation systems, the shopping history and users’ characteristics are being collected and based on these information the system makes proper recommendations (e.g., Amazon, Netflix).

One active research direction is represented by the online communities who rely on the mechanisms offered by trust and reputation systems. A trust and reputation system exploit the users’ experience (resulted from the previous interactions) in order to establish some user-user and user-resource evaluation/trust levels. In the literature there are two categories of algorithms to calculate trust: global and local (also called global trust metrics, respectively local trust metrics). In [1] the advantages and the drawbacks of both approaches are presented.

Currently, most of the approaches focus on the first category. These algorithms aim at quantifying the importance of a user/resource at the community level. These approaches assume that all the users in the community have similar ideas. This assumption lead in the extreme case to a phenomenon described as "tyranny of the majority".

Few approaches are concerned with the development of local trust metrics ([1] [2] [3]). The local trust metrics propagate trust in a limited manner. Each user is connected only to a subset of users/resources. In the extreme case, a segmentation of the society into isolated groups may be achieved.

The challenge consists in designing a system which balances out the two extreme situations.

The work presented in this paper is conducted within a recently proposed system based on local trust mechanisms [4]. A general model of calculating trust and reputation was proposed, which allows a user from the community to have a personalized view on the system. Interesting elements of analysis of these models concern not just users but also groups of users. As shown in the following sections, we focus our research on the techniques which lead to obtaining relevant groups of users, so that the personalized group vision provided by the model to be accurate as well.

The problem is strongly related to community finding within social networks; this is a topic of great interest, intensively studied in the last years. The problem is formally defined as a graph problem: identify groups of vertices within which connections are dense, but between which connections are sparser. Existing methods are based on divisive strategies which compute centrality indices to find community boundaries ([6]) or make use of the hierarchical clustering scheme ([7]). Recently, evolutionary algorithms were proposed to tackle this problem ([8],[9]). The method proposed in this paper, although designed for the particular case of local trust networks, can be applied for general social graphs as well.

Section II describes the trust and reputation system and introduces the problem we analyze: the detection of homogenous groups of users. In section III this problem is analyzed and a solution which is based on Particle Swarm Optimization is described. Section IV presents the experimental results and section V concludes with a brief discussion on the results and some future directions.
II. THE ENGINE RATING OF A TRUST AND REPUTATION SYSTEM

In [4] a personalized evaluation system was proposed, whose goal is to build a flexible and easy way to manage resources in a personalized manner. The purpose of the proposed model is to offer a flexible mechanism to filter irrelevant resources for users.

The system is designed to be used in an online community. Therefore, the main constituents are the users and the resources (their definition is made according to the definition given by T. Berners-Lee, 1998). A relation $\text{worth} : U \times U \rightarrow (0, 5]$ over the set of users is introduced: it is a measure of the level of trust that a user associates to any other user; simplified, $\text{worth}(U_i, U_j)$ may quantify the interest expressed by user $i$ for the resources posted by user $j$. The users assign explicitly one another some ratings which are expressed as real numbers in the interval $(0, 5]$. The five unit-length sub-intervals represent five different levels of trust ranging from "useless/spam" to "exceptional". These values must be used further by the system in order to filter relevant resources for users' queries. Since one user rates explicitly only a subset of the users in community, an algorithm was proposed which propagates these evaluations over the set of users and computes implicit ratings. Such algorithms are called local trust metrics within the Web community.

The current research is focused on identifying homogenous groups of users in this system. For large communities, this analysis is highly motivated:

- Exploratory data analysis by means of clustering is a coherent approach for gathering a general view of the system and consequently an existing clustering algorithm may be introduced. The idea consists of grouping the members of the community into homogenous groups; these groups are further integrated as entities which can express their level of trust among them and can have a reputation. Further, each user will assume the trust statements of the group it belongs to.

Two types of groups can be formed within the presented system: explicit groups and implicit groups. Creating explicit groups is a frequently met phenomenon in the present online communities. The explicit groups are created by users and anyone can create, adhere to or leave such a group.

The implicit groups are to be created by the system based on the user-user evaluations. In this case, using automatic unsupervised techniques, clusters of users will be obtained. Users from the same cluster are supposed to share the same points of view on the community/resources. Consequently, the resources may be clustered in relation to the users who added them in the system. This way of grouping allows the establishment of a quick correspondence between the users clusters and the resources clusters. The motivation for choosing to create such groups is simple: once user A rates highly user B, it would make sense that at some moment it will be interested in the resources posted by user B. Therefore, we get to the level when the system implicitly recommends resources to the user.

III. IDENTIFYING CLUSTERS

This section identifies the main challenges raised by the problem at hand: retrieve homogenous groups of users based on the expressed preferences. The Particle Swarm Optimization paradigm is briefly discussed and the new PSO-based clustering procedure is detailed.

A. Problem statement

In its most general formulation, clustering has a vaguely defined optimum: given a set of data items, "natural" groups should be identified. The problem is formulated in terms of an input consisting of $n$ data items, each described by $m$ numerical attributes. Unfortunately, this representation and consequently an existing clustering algorithm are not straightforward applicable in our case.

In our system, a data item corresponds to a user; each user can be characterized by a vector of length $2n$ containing the ratings it gives to other users in the system and the ratings it receives. Performing clustering based directly on these features would be unfeasible, mainly due to the following drawbacks:

- High dimensionality: the size of the feature space is a multiple of the size of the data set;
- Defining a metric over such a feature space is not straightforward. The similarity between 2 users A and B should be computed with regard to the following assumptions:
  - personal preferences: A is similar to B if A and B rate in the same manner the users within the system;
  - obtained ratings: A is similar to B if they are rated similarly by a large group of users;
  - direct interactions: a high rating given by user A to user B expresses the affinity of user A for user B and therefore, the tendency for user A to adhere to user B’s cluster.

Any existing metric applied to the mentioned feature vectors space would consider the first two clustering criteria but neglect the most important one - the third criterion.
PSO has already been used within the clustering framework. Adopting the idea introduced initially in genetic algorithms approaches for clustering [11] the individuals are (variable-length) strings of cluster centers; data items are assigned to clusters using the nearest-neighbor principle [12]. A survey on Swarm Intelligence techniques applied to clustering can be found in [13]. There exist few approaches with PSO techniques deviating from this centroid-based scheme.

Inspired by dimensionality reduction techniques, Swarm Intelligence algorithms were designed to embed the original data set into a lower-dimensional feature space which preserves the topological relationships among data items. Ant Colony Optimization was used to arrange data items within the cells of a two-dimensional grid, representation well-known from Self Organizing Maps (Kohonen, 1995); a rigorous study on the performance of this approach can be found in [14]. A mapping of the original data set into a two-dimensional Euclidean space is performed using simple PSO rules [15]; although a metric space is employed, the approach is not aimed at generating an embedding of the original data which faithfully preserves the original pairwise distances among data items (as in Multidimensional Scaling approaches); the focus is on identifying clusters through species separation metaphor.

Recently, PSO has been used for enhancing the performance of current clustering algorithms; the clusters are outlined in a pre-processing step which aims at bringing "closer" objects which are likely to belong to the same cluster, while increasing the distance between objects likely to belong to different clusters [16].

C. The embedding procedure

A modified version of PSO is used in our work in order to obtain a two-dimensional representation of the community of users, representation which reflects the interactions/affinities among them. This two-dimensional embedding can be used to visualize the community from a user-oriented perspective: users which give high ratings one another, expressing in this way some common views, will be located close in the two-dimensional Euclidean space. Performing further a simple clustering procedure in this Euclidean space, somewhat homogenous groups of users can be easily identified.

In our approach, the users become particles in a swarm, within a two-dimensional Euclidean space. The mapping procedure is inspired from the PSO algorithm but two essential modifications were introduced.

The motion rules are not governed by a fitness function; the expressed preferences in form of ratings the users give one another guide individual trajectories towards a stable configuration. This is one essential deviation from the basic PSO algorithm: the particles are not assigned unique, unanimously accepted fitness values, based on the position they occupy in the search space. Each particle has its own view on the swarm and its personal "goals", which are expressed as personalized fitness assignments to other particles. There is no global best particle in the swarm or in a subpart of the swarm, but one distinct best particle for each member

In previous work, groups were identified within this system using hierarchical clustering [5]. Usually, the users evaluate and are evaluated by a different number of users. In order to reduce dimensionality, the feature vector for user A is constructed only over the users it interacts with: the users A rates and the users which rate user A. Therefore, the feature vectors have different lengths and a special kind of distance metric derived from the Hausdorff metric for sets was used. Although the results are encouraging, this approach ignores the third criterion detailed above.

In order to eliminate the above mentioned drawbacks, we designed a clustering procedure which is based on the third criterion, incorporates the second criterion explicitly and achieves the first one implicitly. The users are modeled as particles in a PSO algorithm; they move within a two-dimensional space creating a topology that reflects the affinities among them. A simple agglomerative clustering procedure is then applied to cluster the points in the two-dimensional space using the Euclidean metric.

B. Particle Swarm Optimization

PSO is a numerical optimization method, member of the Swarm Intelligence class of algorithms. These paradigms are population-based heuristics inspired from the natural evolution and are part of a larger class of algorithms known as Evolutionary Computation.

Swarm Intelligence systems are typically made up of a population of simple autonomous agents interacting locally with one another and with their environment. Examples of systems like this can be found in nature, including ant colonies, bird flocking, animal herding, bacteria molding and fish schooling.

The PSO model was introduced in 1995 by J. Kennedy and R.C. Eberhart, being discovered through simulation of a simplified social model such as bird flocking [10]. The agents/particles are potential solutions to the problem, their quality being evaluated with the aid of a fitness function. The particles move in the search space with respect to the following rules:

- every particle tends to keep its current direction (an inertia term);
- every particle is attracted to the best position p it has achieved so far (personal best);
- every particle is attracted to the best particle g in population (the particle having the best fitness value);
  - there are versions of the algorithm in which the best particle g is chosen from a topological neighborhood.

Each particle is characterized by a position vector in the search space and a velocity vector which determines its motion. The velocity vector is computed as a weighted sum of the three terms above.

Although PSO was originally conceived as "a method for optimization of continuous nonlinear functions", latter studies showed that PSO can be successfully adapted to solve integer and even combinatorial problems.
of the swarm. However, these personalized fitness evaluations do not counteract the emergence process the general Swarm Intelligence techniques are based on: simple rules describing individual trajectories lead to novel and coherent structures, patterns and properties during the process of self-organization (Goldstein 1999).

Another deviation from the PSO basic scheme is a change in motion rules. Except the attraction towards the best particle, the remaining two rules are re-defined. The procedure is detailed below.

In an initialization step, each user is assigned a randomly-generated point in a two-dimensional space; these points are to be further considered as position vectors of the particles in a swarm. Then, an iterative process aims at organizing the particles, such that the users with “strong interactions” will be matched to close points in the Euclidean space. In the sequel, the terms particle/user are interchangeable: both denote individuals in a swarm, characterized by a position vector in a two-dimensional space and a velocity vector; an $n \times n$ matrix stores the pair-wise affinities in the form of real numbers in the interval $[0,5]$, designating the ratings the users/particles give one another. The rules which guide the particles are described next:

- Each particle $x_i$ moves toward the preferred particles. To this end, a weighted centroid is computed for the $k$ particles highest rated by particle $x_i$. Considering these rates as fitness values, the centroid can be viewed as the social factor in PSO - the tendency to move towards the best particle in the neighborhood. The centroid is computed as:

$$g_i = \frac{\sum_{j=1}^{k} \text{mark}(i,j) \cdot x_j}{\sum_{j=1}^{k} \text{mark}(i,j)}$$

This move, based only on the direct interactions, leads to indirectly fulfilling the first requirement for a meaningful clustering. Two users which give identical ratings move toward the same point in the two-dimensional space; therefore, the distance between them is minimized.

- Each particle moves towards the particles that rate it. To this regard, a weighted centroid $\overrightarrow{g_i}$ is computed as previously, from the particles that rate particle $x_i$: This rule is necessary in order to accomplish the second requirement for clustering: two users which are rated identically should reside close within the two-dimensional representation.

- Each particle moves away from $k$ neighbors it did not rate or it assigned a low rate and vice versa. The neighborhood is defined with the aid of a threshold: the maximum distance in the two-dimensional space within which the particles interact. The centroid over $k$ random particles in the neighborhood is thus computed (denoted as $\overrightarrow{g_i}$).

The formulas used to update the particle $i$ at iteration $t$ are:

$$v_i^t = w_1 \cdot (\overrightarrow{g_i}^t - x_i^{t-1}) + w_2 \cdot (\overrightarrow{g_i}^t - x_i^{t-1}) + w_3 \cdot (x_i^{t-1} - \overrightarrow{g_i}^t)$$

$$x_i^t = x_i^{t-1} + w_{\text{max}} \cdot v_i^t$$

The parameters and the stopping criterion are empirically determined (see section IV.A).

The PSO procedure is followed by a single link agglomerative procedure which can be stopped when a fixed number of clusters is achieved or when the distance between clusters exceeds some threshold. Since the two-dimensional representation obtained with PSO can be visually analyzed, the number of clusters can be specified. Anyway, the use of k-Means algorithm is not recommended: as shown in the experimental section, the resulted clusters have different shapes and volumes and even some isolated points may be obtained.

IV. EXPERIMENTS

A. Parameters settings

In order to obtain a first empirical setting of PSO parameters, small problem instances were created; the resulted two-dimensional maps are validated by analyzing the ratings table. One such instance is represented in Figure 2; two maps created using different initializations are illustrated in Figure 3.
Fig. 3. Two different mappings corresponding to different initialization

\[ w_{\text{max}} = 0.15. \]

A threshold concerning the ratings was used: only ratings greater than 2 denote strong affinities and consequently attraction forces.

For small instances, as the one illustrated in Fig. 2, the algorithm reaches a stable configuration after 50 iterations. For larger communities, more iterations are required until a stable configuration is reached. As a halting criterion, an average over velocity vectors may be computed and compared to a scalar threshold: small velocities correspond to stable configurations. We look for the number of iterations after which no major changes on the two-dimensional map are observed.

As one can observe in Figure 3, the initial configuration influences the orientation of clusters and, to some extent, their relative positions; on the other hand, the same clusters are detected in repeated runs (different initializations) and within each cluster the components are arranged under the same topology.

These first empirical results are illustrative for the convergence of the algorithm: repeated runs involving different initial configurations lead (after a limited number of iterations) to similar stable configurations. The similarity between configurations is measured in terms of detected clusters: the number of clusters, the constituents of each cluster and even the shape of the clusters. All these elements illustrate topology preservation.

A more in-depth analysis is necessary in order to verify the agreement between the resulted mapping and the graph of ratings. In Fig. 3, due to the single-directional high ratings, one linear cluster is formed over particles 1, 3, 8, 11 and 10. Particles 5 and 13 end-up close together due to bi-directional (reciprocal) high ratings; following them at greater distance particles 6 and 7 are situated. Due to the lack of any evaluations/ratings, particles 2, 9 and 14 are isolated in different regions of the map.

### B. Data generator

Once empirically tuned on synthetic small data sets, the method needs to be tested on larger (real) data sets. Unfortunately, existing systems/comunities do not make public the data and no benchmark is available. Previous studies on local trust metrics report some tests on data extracted from Epinions community (Massa, Avesani, 2007). Ziegler and Lausen, (2004) created synthetic data for tests.

In order to simulate real communities, we designed a somewhat complex data generator. It takes as input an xml file which can be customized to generate different categories of explicit evaluations. The following parameters are introduced:

- **users** - the number of users in the community;
- **goodUsers** - an approximation of the number of “good” users in the community; that is, the number of users who receive the majority of good evaluations;
- **minMarksCount** - the minimum number of evaluations realized by an entity; for a better observation of the dynamics of spreading the trust each user must interact with the community;
- **goodMarksThreshold** - the threshold which separates the good evaluations from the worse ones; this parameter is further used in the clustering procedure;
- **userRatingsDensity** - a percentage which represents the density of the community’s network: how many explicit evaluations are made in the system (from a total of \( n^2 \) possibilities);
- **goodUserMaxDivergence** - a percentage representing the number of bad marks that can be awarded to an user already designated as being “good”.

### C. Experimental settings

As shown in section IV A, the final mapping is strongly dependent on the initial configuration. Due to random initializations, the stability of the method must be investigated. This analysis is compulsory for probabilistic heuristics and generally consists in computing the variance of solutions obtained in repeated runs.

As the main goal of the method is to identify meaningful clusters, we study the stability of the method with regard to the detected clusters.

On small problem instances, as the one presented in section IV A, the results are easy to analyze visually. For larger problem instances the analysis we conducted consists of the following steps:

- run several times PSO-mapping on the same problem instance with random initializations;
- apply a deterministic clustering algorithm on each mapping;
- measure the similarity between the clustering solutions (the partitions) and compute an average.

To perform clustering in the two-dimensional space we designed a single link agglomerative clustering procedure; the procedure stops when the distance between classes exceeds a given threshold.
As measure of similarity between two partitions we used the Adjusted Rand Index which is an external validation measure intensively used in clustering literature.

Given two partitions C and U, the Rand Index (RI) records the following information:
- \(a\) - the number of pairs of data items that are placed in the same cluster in C and in the same cluster in U
- \(b\) - the number of pairs of objects in the same cluster in C but not in the same cluster in U
- \(c\) - the number of pairs of objects in the same cluster in U but not in the same cluster in C
- \(d\) - the number of pairs of objects in different clusters in both partitions.

The Rand Index is computed as \(\frac{(a + d)}{(a + b + c + d)}\). The quantities \(a\) and \(d\) can be interpreted as agreement, while \(b\) and \(c\) as disagreement.

We require the similarity measure to take values close to 0 for two random partitions and value 1 for identical partitions. In order to achieve the first criterion, a normalization by the results expected from random data is needed. The Adjusted Rand Index, which incorporates such normalization, can take on values in a wider range, thus increasing the sensitivity. In general, the normalization corrects the estimated absolute degree of quality. The Adjusted Rand Index is given by the formula:

\[
ARI = \frac{\sum_{ij} C_{N_{ij}}^2 - \left(\sum_i C_{N_{i-}}^2 \cdot \sum_j C_{N_{-j}}^2\right) / C_N^2}{\frac{1}{2} \left(\sum_i C_{N_{i-}}^2 + \sum_j C_{N_{-j}}^2\right) - \left(\sum_i C_{N_{i-}}^2 \cdot \sum_j C_{N_{-j}}^2\right) / C_N^2}
\]

where \(N_{ij}\) is the number of data items both in cluster \(i\) in \(C\) an in cluster \(j\) in \(U\), \(N_{i-}\) is the number of data items in cluster \(i\) in \(C\) and \(N_{-j}\) is the number of data items in cluster \(j\) in \(U\).

Problem instances were generated for different parameters of the data generator. Tests were performed on problem instances consisting of 20, 50 and 100 users.

In each case, 5 problem instances were generated; PSO and the subsequent clustering procedure were applied five times on each problem instance. The similarity measure was computed for each pair of the resulted partitions and the average over all pairs is reported.

### D. Experimental results

Table 1 presents the experimental results over several classes of problem instances generated for different settings of the data generator. The parameters of the data generator are specified in the following order: \texttt{users-goodUsers-userRatingsDensity}. The minimum number of evaluations given by a user was set to 1. In all experiments the value for \texttt{goodUserMaxDivergence} was set to 10% ; this defines a user as being "good" if more than 90% of the ratings it receives are greater than \texttt{goodMarksThreshold} parameter value.

As the main purpose of the generator is to simulate real online communities, the average number of obtained clusters is computed in order to study their dynamics.

The high ARI values reported in Table 1 indicate coherent/similar partitions obtained at repeated runs, which further suggest stable and similar mappings produced by the PSO procedure. Anyway, the mapping algorithm is influenced to some extent by the initial configuration and the stability of the algorithm seem to depend on the parameters of the problem.

As expected, the stability of the mapping procedure based on PSO generally decreases with the size of the problem, as the number of possible configurations grows exponentially.

One important parameter which acts in the opposite way is the ratings density. When there are many evaluations in the system, complex interactions dictate the dynamics of the system and, consequently, of the swarm in the mapping procedure. The PSO particles are subjected to more restrictions regarding their motion in the two-dimensional space. To reach a stable configuration a larger number of iterations is needed but the outputs are less sensitive to the initial configuration.

Increasing the number of good users in the system, some cluster nucleus appear and well-defined clusters are to be formed. Consequently, the particles concentrate in dense clusters where the repulsion forces are minimized and the algorithm is more stable for such configurations.

Some general conclusions regarding the influence of the mentioned parameters on the number of clusters may be drawn.

The number of good users in the system has a great influence on the number of clusters. In order to coexist more good users in the system, several users have to rate them high. This lead to few larger clusters.

The same influence is observed for the \texttt{userRatingsDensity} parameter: few explicit ratings in the system determine sparse configurations and consequently, many small clusters.

The method can be used to study further the dynamics of the system. Many scenarios can be imagined. For example, one interesting case study concerns the impact of a new user’s evaluations on the system. The structure of clusters and, furthermore, a visualization of the embedding produced with the proposed algorithm are useful tools for such analysis.

<table>
<thead>
<tr>
<th>PROBLEM INSTANCE</th>
<th>ARI</th>
<th>NUMBER OF CLUSTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-15-0.3</td>
<td>0.9611</td>
<td>3</td>
</tr>
<tr>
<td>20-10-0.3</td>
<td>0.8555</td>
<td>8</td>
</tr>
<tr>
<td>50-35-0.5</td>
<td>0.9376</td>
<td>3</td>
</tr>
<tr>
<td>50-20-0.5</td>
<td>0.8853</td>
<td>4</td>
</tr>
<tr>
<td>50-35-0.3</td>
<td>0.7969</td>
<td>4</td>
</tr>
<tr>
<td>50-20-0.5</td>
<td>0.8712</td>
<td>7</td>
</tr>
<tr>
<td>50-35-0.1</td>
<td>0.7994</td>
<td>12</td>
</tr>
<tr>
<td>50-20-0.1</td>
<td>0.8755</td>
<td>16</td>
</tr>
<tr>
<td>100-65-0.1</td>
<td>0.7832</td>
<td>14</td>
</tr>
<tr>
<td>100-50-0.1</td>
<td>0.7634</td>
<td>17</td>
</tr>
<tr>
<td>100-35-0.1</td>
<td>0.6323</td>
<td>22</td>
</tr>
<tr>
<td>100-35-0.3</td>
<td>0.8251</td>
<td>4</td>
</tr>
</tbody>
</table>

**TABLE I**

Average Adjusted Rand Index and the average number of clusters for different classes of problem instances.
E. Applications to social networks

With minimal modifications, the method can be applied to identify communities in social networks.

To study its performance, the American Football data set introduced in [6] is used. The data set can be represented as a graph consisting of 115 nodes and 616 edges: the nodes represent football teams and the edges represent regular season games between the two teams they connect. The teams are divided into 11 "conferences". The teams play an average of about 7 intra-conference games and 4 inter-conference games.

We apply the PSO/clustering method described previously with the aim of identifying the "conferences". In order to benefit from the previous scenario, the original problem graph is transformed into a weighted digraph: each edge is replaced by two opposite arcs and equal weights are assigned to all arcs. Under this representation, the second term in the PSO-mapping algorithm is identical to the first term and therefore, may be eliminated. Equal weights (0.5) were set for the two remaining terms. A run for the PSO-mapping procedure consists of 500 iterations; the subsequent single link clustering procedure is stopped when the cluster inter-distance exceeds 0.75 (3/4 reported to the activation threshold for the repulsion force which is set to 1).

In order to compare the results obtained by our method with the actual constitution of the "conferences", an average over 10 runs for the Adjusted Rand Index was computed; the obtained value of 0.7763 indicate a close match between the generated partitions and the actual partition. The divergence is mainly due to a higher number of detected clusters (15 on average) and only to a little extent is due to wrong allocations. To illustrate this, the mapping and the resulted clusters for a low-performance run (ARI=0.71) are presented in Fig. 4.

The mapping takes the regular form of rectangular lattice. Clusters 0, 1, 4 and 8 are matched perfectly to the real partition. Each one of the actual classes 2, 3, 6 and 9 are divided by our mapping into two adjacent clusters. One team from class 11 was misclassified. Regarding class 7, the two clusters for a low-performance run (ARI=0.71) are presented in Fig. 4.

The obtained results indicate that the presented method is an appropriate approach for community detection in social networks.

V. CONCLUSION AND FUTURE WORK

The main aim of this paper is to propose a method to get insights into the structure and dynamics of online communities which operate on mechanisms of trust. An embedding of the community into a two-dimensional Euclidean space, which reflects affinities among users, allows by means of simple clustering procedures to identify natural groups. As a basis of this method, Particle Swarm Optimization is employed: the simple individual rules which lead to self-organization in complex systems (the emergence process) fit perfectly to this problem.

The identification of cohesive communities by means of clustering procedures is a key process in social network analysis. The current approach can be applied to general social graphs as well.

As feature work, the mapping/clustering procedure will be extended by introducing direct resources evaluations in the current local trust network.

REFERENCES

Fig. 4. Mapping for the American College Football network; the teams are represented as points in the two-dimensional space; well-defined clusters are identified; the clusters are specified in brackets followed by the actual membership of the teams.