Measuring Similarity between Sets of Overlapping Clusters

Mark K. Goldberg, **Mykola Hayvanovych** and **Malik Magdon-Ismail**
Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180
{goldberg,hayvam,magdon}@cs.rpi.edu

***Abstract*—The typical task of unsupervised learning is to organize data, for example into clusters, typically disjoint clusters (eg. the K-means algorithm). One would expect (for example) a clustering of books into topics to present overlapping clusters. The situation is even more so in social networks, a source of ever increasing data. Finding the groups or communities in social networks based on interactions between individuals (a measure of similarity) is an unsupervised learning task; and, groups overlap – an individual can be a chess player and a violin player, in which case he would interact with members of both these groups.**

The problem we address is not that of finding the overlapping clusters, but of *comparing* two sets of overlapping clusters. Such a task is the basis for comparing two different clusterings, which is important for comparing algorithms with each other or with a ground truth. From the social network point of view, we are particularly interested in quantifying social group evolution – how much the social group structure of a social network changed – by comparing the set of groups at consecutive time intervals.

There is significant prior work on comparing sets of disjoint clusters (partitions). When overlap is allowed, the problem becomes considerably more complex owing to the possibility of degeneracies, which we illustrate through examples. We describe three novel definitions of the distance between collections of potentially overlapping sets, and present algorithms for computing those distances. We test our algorithms on diverse data sets: collections composed from social groups in Twitter, Blogosphere and Enron Email data.

I. Introduction

A generic task in unsupervised learning is to organize data into clusters. In the online world of social interactions, the examples are many: organize books into topics based on similarities between books (for example two books are similar if the same person read or purchased both); arrange the members of a social network (eg. Twitter or the LiveJournal blogosphere) into communities based on who interacts with whom. Such a categorization of the data should ideally result in overlapping clusters (it is possible to have a science fiction romance, or a person who belongs to a chess group and a religious organization, etc.).

Our motivation comes from social networks, which are a source of abundant data (see Figure 1). Figure 1 shows example social networks at two consecutive time periods; the networks have been clustered into communities. The task of clustering data into overlapping groups is indeed a challenging one, which is far from solved; however, many approaches to solving this problem exist, ranging from latent Dirichlet analysis to local optimality based methods, [2], [12], [11], [10], [5]. It is not our goal to address this problem of finding clusters, and indeed, we take as "solved", the task of identifying the overlapping communities. So, for example, the groups identified in Fig. 1 are given.

We address an auxiliary problem. In dynamic social networks (eg. Figure 1), which shows the behavior of the Iranian Blogosphere, it is important to be able to measure the rate of change of groups over time and to quantify the society evolution in general. As can be seen in these examples, some of the groups grow rapidly over the period of one year, new groups appear, while there can also be groups which merge (eg. in 2008 the groups "twelver" and "religious youth" evolve into the group "CiberShia" in 2009 [6]). So, by how much has the community structure changed through this evolution? To answer this question, we need to be able to measure the distance between sets of overlapping sets. While our motivation lies in quantifying the rate of change in community structure, the task of comparing two sets of clusters has numerous applications, the most obvious being to compare the results of a clustering algorithm to a ground truth clustering, or to compare two different clustering algorithms.

When it comes to partitions (clusterings into disjoint clusters), numerous approaches exist, both for comparing hard partitions and soft or fuzzy partitions. These methods can be applied in a variety of settings including to subspace clusterings, partial clusterings and hierarchical clusterings, see for example [3], [13], [8].

The situation is considerably more complicated when dealing with clusterings of *overlapping* clusters. Lets first define the problem more concretely. The general goal is to measure the distance between two sets of clusters. A *clustering* is a set of clusters. The clusters in a clustering may overlap and need not be a cover.

We note that a related problem to comparing discrete clusterings is to compare two probability densities (for example two Gaussian mixture densities) which can be viewed as identifying "clusters" in a metric space. Methods for comparing probability distributions exist (for example the Kullback-Leibler distance). Such mixture distributions can be viewed as identifying cluster centers, and then assigning probabilistic cluster membership to the data points - the clusters are still disjoint, it is just that the membership is uncertain. This is a different problem from discrete overlapping clustering, where the task is to create *overlapping* clusters with boolean memberships. To make the discussion more concrete, we need some notation.

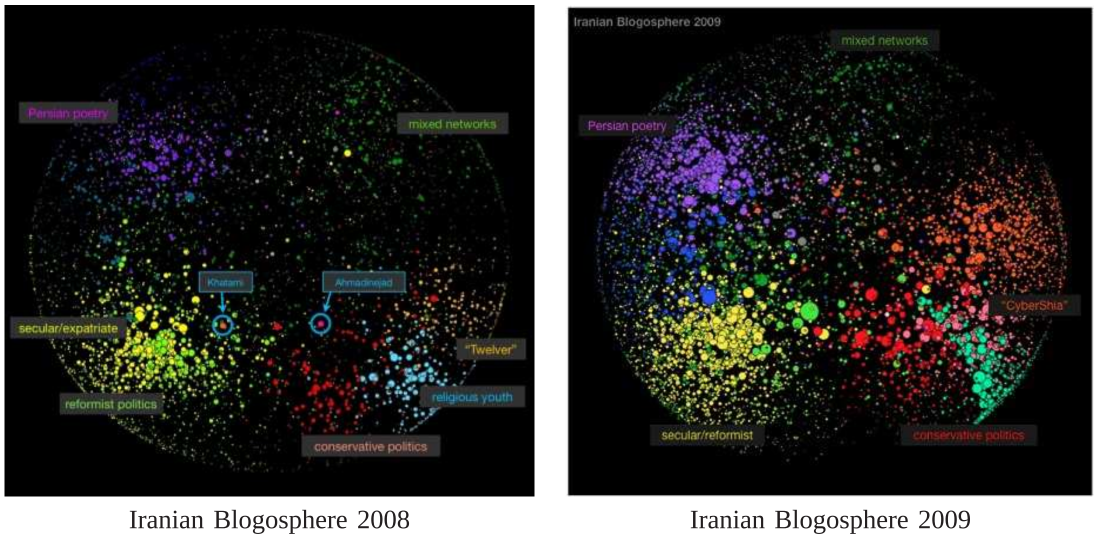

Fig. 1. Evolution of part of the Iranian Blogosphere over the period of 2008 and 2009. Note: each dot is a Blog, the size of the dot represents the number of other blogs that link to it, a measure of its popularity. The position of each dot is a function of its links with its neighbors (same topic Blogs tend to gather together and form "clusters").

Let $C_1 = \{S_1, S_2, \ldots, S_n\}$ and $C_2 = \{S'_1, S'_2, \ldots, S'_m\}$ be two clusterings of size n and m respectively, where S_i and S'_j are the individual clusters; we assume that a cluster does not contain duplicates. The goal is to compute $d(C_1, C_2)$, the distance, between the clusterings C_1 and C_2. The algorithm should be efficient, and the distance should reflect the "intuitive" distance between the cluster structures that C_1 and C_2 represent. The crux of the difficulty is illustrated by the example in Figure 2.

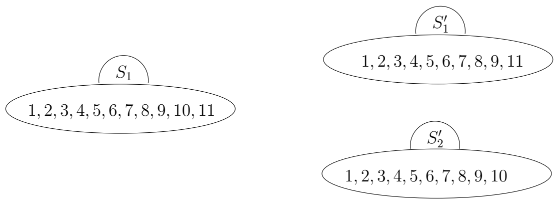

Fig. 2. Two *overlapping* clusterings of the same data: on the left, one cluster with 11 members; on the right, 2 clusters with 10 members each.

On the right, there is essentially one big cluster, $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (the intersection) and the cluster-structure on the right essentially says that items 10 and 11 have slightly different relationships to this big cluster, hence two clusters; on the left, all elements are in the single big cluster. So essentially these two clustering are representing very similar structures – one big cluster, with minor perturbations. However, the significant overlap between clusters will lead to serious problems for naive approaches to quantitatively compare these two simple clusterings.

A simple way to define the distance between two clusterings is to consider the number of changes (moves) necessary to convert one clustering into the other; or, alternatively to solve a weighted bipartite matching problem to assign the clusters in the first clustering to clusters in the second clustering (without loss of generality, we can assume the two clusterings are of the same size by adding empty clusters to the smaller clustering). These problems can be solved by a number of approaches such as: the *Hungarian* algorithm [9], the *Max-Flow* approach or by formulating a Linear Programming problem. All of these approaches try to "convert" one clustering into the other by finding the lowest cost matching, and are reasonably efficient (cubic in the clustering sizes). Unfortunately, all of the algorithms mentioned above do not account for the potential overlap between the groups and thus are not suited well to work with clusterings found in social networks.

This naive approach works quite well for partitions, but if we try to apply such a naive approach to the example in Fig. 2 we run into a problem. First we need to add an empty cluster to the left clustering. Then, to convert the left clustering to the right would require 11 moves, one move per element. This suggests that the two clusterings are significantly different, counter to the intuitive expectation. Thus, while many of the known approaches perform well for clusterings of equal size and little or no overlap, they will fall short when it comes to measuring the distance between clusterings when such overlap degeneracies exist.

Our Contribution. We propose tree different methods for comparing clusterings which can handle overlap among clusters. The *Best Match* and *K-center* approaches, intuitively compute the relative number of moves necessary to transform C_1 into C_2, but in a way which takes into account overlaps; The *Interaction Probability* approach first assumes an interaction model and computes the probability that two individuals interact given a clustering. The problem is then reduced to comparing two sets of probabilities, which can be done using any standard vector norm. We demonstrate our measures on Twitter, Blog and email networks, and compare with a very recent method which can also tolerate overlaps [7].

II. Comparing Clusterings

We present three methods for comparing clusterings. In all cases, the input is the two clusterings $C_1 = \{S_1, S_2, \ldots, S_n\}$ and $C_2 = \{S'_1, S'_2, \ldots, S'_m\}$, and the output is a "distance", $d(C_1, C_2)$. In some cases, one can choose d to have the standard metric axioms, however this will not generally be so. This is not a big problem because we are not interested in geometrically manipulating clusterings, just measuring similarity.

A. Best Match

We start with the intuitively simplest *Best Match* algorithm, which takes as input the two clusterings C_1, C_2 and a set difference measure $d(S, S')$. The function d takes as input two sets and computes a "distance" between them. The best match algorithm determines how well the clusterings represent each other. Typically, $d(\cdot,\cdot)$ will have all the properties of a distance measure (positivity, symmetry and the triangle inequality), but this is not a requirement.

Specifically, for each cluster $S_i \in C_1$, we can compute its best representative in C_2: a cluster $S' \in C_2$ such that $d(S_i, S') \le d(S_i, S'_j)$ for all $S'_j \in C_2$. Thus,

$$d(S_i, S') = \min_{j=1,\ldots,m} d(S_i, S'_j).$$

We compute how well C_2 represents C_1 by summing distances from each member of C_1 to its respective best representative in C_2. We make this distance symmetric by also summing the distances from every member of C_2 to its corresponding best representative in C_1. This gives the final symmetric measure

$$d(C_1, C_2) = \sum_{i=1}^{n} \min_{j=1,\ldots,m} d(S_i, S'_j) + \sum_{j=1}^{m} \min_{i=1,\ldots,n} d(S'_j, S_i).$$

One could normalize this distance by the number of clusters, $|C_1| + |C_2|$ to get a per-cluster distance, or by the number of distinct elements in the clusters to get a per-element distance. To fully specify the method, one needs the set difference function $d(S, S')$. The computational complexity of this method is given by the complexity of computing the distance d between all pairs of clusters (S_i, S'_j).

1) Examples: We present some natural choices for $d(S, S')$. The first is simply the the number of moves necessary to convert S into S':

$$d_1(S, S') = |S| + |S'| - 2|S \cap S'|.$$

One advantage of this measure is that it is a sub-modular set function, which leads to some useful algorithmic properties. Normalizing by $|S \cup S'|$, we get the set difference measure:

$$d_2(S, S') = 1 - \frac{|S \cap S'|}{|S \cup S'|}.$$

In a recent development, independently of our work, an entropy based measure is introduced to compare clusterings allowing for overlap between clusters [7]. This is the only other quantitative approach we have seen which can tolerate significant cluster overlaps. We denote this measure the *Entropy* based measure, and use it as a benchmark to compare with the methods we propose. This Entropy measure is a special case of the best match algorithm with a particular choice of set distance function, where $d(S, S')$ is based on how much information the set S' conveys about S – it is an asymmetric distance measure. Specifically, assuming a random model for constructing S and S', given the size of the intersection $|S \cap S'|$ and the sizes $|S|$, $|S'|$, $d(S, S')$ captures how much uncertainty remains about S, given S'. The notion of conditional entropy captures exactly this notion, hence,

$$d(S, S') = H(S|S').$$

The independent random model for constructing S, S' gives:

$$P_x = P[x \in S] = \frac{|S|}{N}, \text{ and} P_y = P[y \in S'] = \frac{|S'|}{N},$$

where N is the number of elements. We therefore have:

$$\begin{aligned} P_{11} = P[x \in S, y \in S'] &= \frac{|S \cap S'|}{N}, \\ P_{10} = P[x \in S, y \notin S'] &= \frac{|S| - |S \cap S'|}{N}, \\ P_{01} = P[x \notin S, y \in S'] &= \frac{|S'| - |S \cap S'|}{N}, \\ P_{00} = P[x \notin S, y \notin S'] &= 1 - \frac{|S| + |S'| - |S \cap S'|}{N}. \end{aligned}$$

The distance is given by

$$d(S, S') = H(X|Y) = H(X, Y) - H(Y),$$

where the joint and marginal entropies aregiven by:

$$\begin{aligned} H(X, Y) &= -\sum_{a,b=0}^{1} P_{ab} \log P_{ab}, \\ H(Y) &= -P_y \log P_y - (1 - P_y) \log(1 - P_y). \end{aligned}$$

There is one caveat in this measure, which is that $H(X|Y) = 0$ if S is the complement of S', because knowing S' completely determines S. For this reason the authors in [7] suggest the heuristic of using $H(X)$ for such situations, i.e. when $-\sum_a P_{aa} \log P_{aa} < -\sum_{a \ne b} P_{ab} \log P_{ab}$.

Observe that all these examples of set distance functions only require knowledge of the sizes of the clusters and the sizes of the pairwise intersections. This need not be the case in general.

B. K-center

Unlike the *Best Match* approach in the previous section, which works on a cluster by cluster basis, the *K-center* method first constructs a smaller representation of the two clusterings; the smaller representation is of size K. Then algorithm then measures the distance between these two smaller representations. Again, we have inputs C_1, C_2 and the function $d(S, S')$.

We first discuss how to obtain this smaller representation, which is a K-center. Given (say) $C_1 = (S_1, \ldots, S_n)$ and K, a K-center $Center_K(C_1)$, is a subset of C_1 of size K which minimizes:

$$\sum_{i=1}^{n} d(S_i, Center_K(C_1)),$$

where

$$d(S_i, Center_K(C_1)) = \min_{j=1,\dots,n}(d(S_i, S_u)).$$

Intuitively $d(S_i, Center_K(C_1))$ is the distance from S_i to the closest S in $Center_K(C_1)$. The K-center captures as much information/structure of the original clustering as possible while using at most K sets of the original clustering.

In general, computing an optimal K-center is NP-hard. We give a greedy heuristic, which is known to produce an $(e-1/e)$ approximation [4] when using the set difference measure $d_1(S, S')$ which was described in the previous section. This approximation guarantee is due to the sub-modularity mentioned earlier. Using $d_2(S, S')$, we lose submodularity, hence we lose the approximation guarantee, however in practice the greedy heuristic works well here too.

The greedy heuristic starts with an empty K-center, and at each step adds the set which is furtheest away from the center, until K sets have been added. An efficient implementation of this algorithm maintains at each step the distance of each set from the current working K-center. Initially, these distances are given by $|S_i|$ (for the distance measure d_1), so the greedy algorithm starts by adding the largest set. [1] One then updates the distances of each element to the K-center in $O(n)$ updates, where each update computes the distance from each set to the set just added; to obtain the distance of a set to the new working K-center, one simply takes the minimum of the distance to the set just added and the previous distance.

The full algorithm to compute $d(C_1, C_2)$ takes the two clusterings C_1 and C_2 and K; one first finds the two K-centers $Center_K(C_1)$ and $Center_K(C_2)$ of C_1 and C_2 respectively, each of size at most K. After obtaining $Center_K(C_1)$ and $Center_K(C_2)$, we can use one of the naive measures of distance (*Best Match* or the Hungarian algorithm) to compute the distance between the two K-centers. Typically the K-centers will tend to have low overlap (especially for small K) hence the naive methods will work here as well.

Again, we can normalize by the total number of distinct elements or the number of clusters. The *K-center* algorithm performs well on clusterings of different size and with overlapping sets. It can be sensitive to clusterings of sets with size that highly deviates from an average set size when using d_1 as the distance measure, as the large sets will be the sets included in the K-centers. This situation is improved by using d_2.

C. Interaction Probability

The *Interaction Probability* approach assumes an underlying interaction basis for the clusters, which is especially well motivated in social networks, but would be applicable in other settings. Specifically, a cluster of nodes should interact intensely among themselves and not as intensely with outsiders. Thus, given two clusterings of the same nodes, they represent two sets of pairwise interaction probabilities. We can thus compare the two clusterings by comparing the induced set of interaction probabilities. The advantage here is that clusterings are complex, however there are only $N(N-1)$ (directed) interaction probabilities, so we reduce the problem to that of comparing two sets of probabilities. The disadvantage is that one has to assume an interaction probability model, and the result can depend on the assumed model. We will present a particular interaction probability model based on some simple intuitive assumptions.

Given the two clusterings C_1 and C_2 first we compute the underlying interaction probability graphs G_1 and G_2 respectively. We construct the graph of communication probabilities G_1, by computing for each pair of elements i, $j \in C_1$ the probability of an interaction between the two elements i, j, denoted P_{ij} based on the following simple model. First, we assume that a node will (on average) interact with P_g nodes in a cluster that it belongs to. So, if i, j are in the same cluster S, then i will interact with j with probability

$$P_S^{int}(i, j) = \frac{P_g}{|S|}.$$

The node i has such a probability to interact with j for every cluster they are both members of - we call this *cluster based interaction*. There is also a possibility of random interaction. We assume that a node will on average have P_e random interactions. So the probability of a random interaction from i to j is

$$P^{ext}(i, j) = \frac{P_e}{N}.$$

The constants P_e and P_g are user specified, and N is the total number of elements (the number of nodes); P_e represents the extent of extra-cluster communication and P_g the intensity of intra-cluster communication.

We can now compute the probability of no interaction, assuming that the random (extra cluster) and intra-cluster interactions are all independent: $(1 - P^{ext}(i, j))$ is the probability of i and j do not communicate randomly; similarly $(1 - P_S^{int}(i, j))$ is the probability of i and j not communicating within group S. The product of above expressions reflects the total probability of i and j not communicating both internally and externally. Accounting for the fact that i, j may be in multiple groups together in a clustering, we have

$$P_{C_1}(i, j) = 1 - (1 - P^{ext}(i, j)) \cdot \prod_{k=1}^{n} (1 - P_{S_k}^{int}(i, j)).$$

We have a similar expression for $P_{C_2}(i, j)$. Once we have constructed $P_{C_1}(i, j)$ and $P_{C_2}(i, j)$ for all pairs, we can find the similarity/distance between them by using any measure for two sets of probabilities, for example, the average L_2 distance or the Kullback - Leibler distance between the probabilities. We denote these measures M_{L_2}, M_{KL} respectively.

[1]For the distance measure d_2, the initial distances are all 1 and so the greedy heuristic starts by adding a random set.

$$M_{L_2}(G_1,G_2) = \sqrt{\frac{1}{N^2}\cdot\sum_{\forall(i,j)}(P_{C_1}(i,j)-P_{C_2}(i,j))^2}$$

$$M_{KL}(G_1,G_2) = -\frac{1}{N^2}\sum_{\forall(i,j)}(P_{C_1}(i,j)\cdot\log(P_{C_2}(i,j)) + (1-P_{C_1}(i,j))\cdot\log(1-P_{C_2}(i,j)))$$

	C_1 - C_2	C_2 - C_3	C_3 - C_4
Best Match	0.986	0.971	0.969
K-Center	0.981	0.980	0.977
M_{L_2}	0.01191	0.01171	0.01138
Entropy	0.998	0.998	0.997

	C'_1 - C'_2	C'_2 - C'_3	C'_3 - C'_4
Best Match	0.17	0.26	0.21
K-Center	0.14	0.26	0.2
M_{L_2}	0.07	0.08	0.07
Entropy	0.197	0.304	0.241

Fig. 4. The rate of change of the clusterings in Blogosphere over the period of four weeks on the left and The rate of change of the clusterings in the Enron organizational structure from 2000 - 2002 on the right.

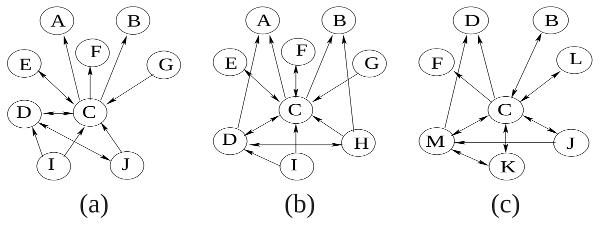

Fig. 3. Evolution of part of the Enron organizational structure over the periods (a): Sept. 2000 - Sept. 2001; (b): Mar. 2000 - Mar. 2001; (c): Sept. 2001 - Sept. 2002. Note: actors B, C, D, F present in all three intervals. Here is who they are: B - T. Brogan, C - Peggy Heeg, D - Ajaj Jagsi and F - Theresa Allen.

	Best Match	*K-Center*	M_{L_2}	*Entropy*
C''_1 - C''_2	0.857	0.91	0.0144	0.9

Fig. 5. The rate of change of the clusterings in Twitter network over the period of twenty weeks.

III. Experimental Results

A. Tracking the Evolution of Community Structure

Our main application is to the quantitative measurement in the change of community structure in social networks. As mentioned earlier, we take as given the communities of a social network at given time periods. Some typical methods for obtaining such communities are to aggregate the communications over a sliding time window, and use some overlapping clustering algorithm to obtain the communities over that time window, or to directly detect communities in the streaming communications [2], [1]. We show an example evolution of one such discovered cluster in the Enron email data in Fig. 3. These results were taken from [1]. One by product of the best match algorithm is that the best match at the next time step to a cluster in the previous time step serves to identify the cluster's evolution. We ran three experiments to compare the rates of community change in Enron, the LiveJournal Blogosphere and Twitter.

We compared 4 methods for measuring community evolution. *Best Match* is the best match approach with d_2 as the set distance measure, and the Entropy method is the best match approach with the entropy based distance measure from [7]. We also compared the K-center method with $K = 0.75(|C_1|+|C_2|)/2$, using the greedy heuristic with d_2 to compute the K-centers; we then used Best Match to compare the K-centers. Finally, we used the interaction probability based method ML_2, with $P_e = P_g = 1$.

We normalized all methods except M_{L_2} by the number of clusters to obtain a difference per cluster; M_{L_2} is normalized by the number of possible pairwise interactions, which is $O(N^2)$.

a) Blogs: We used blog communications over 4 consecutive weeks to construct four clusterings C_1, C_2, C_3 and C_4. Each consecutive clustering was constructed one week later than the previous, and for a consecutive pair of clusterings, the number of distinct nodes in the network was about $N = 13,000$.

We can see in the Fig. 4 that the algorithms imply that the rate of change of groups in the Blogosphere is high and the groups change very dynamically from one week to another.

b) Twitter: We repeat this experiment with the Twitter data. We have two clusterings C''_1 and C''_2, which were computed on two coinciding ten week periods. For these graphs, $N \approx 2700$.

We used our similarity measures to track the rate of change in the network, shown in Fig. 5. Notice that the rate of change in the Twitter network over a *10 week period* is lower than the rate of change in Blogs over a *1 week period*.

c) Enron: We repeat this experiment with the Enron data. On each window we obtained the clusterings C'_1, C'_2, C'_3 and C'_4. For these clusterings, $N \approx 100$. These clusterings were for the network 6 months apart in time.

Next we used our similarity/distance measures to track the rate of change in the network. Fig. 4 illustrates the rate of change of clusterings. Notice that the rate of change in the email networks over a *6 month period* are significantly lower than the rate of change in Blogs over a *1 week period*. Blogs are a significantly more dynamic social network which should be no surprise.

We note that Blogs, Enron and Twitter are three completely different social networks. Enron represents a company organizational network, which has the underlying hierarchy of command that is unlikely to change quickly over time; the Twitter is a dynamic social network, which grows and changes quickly over time; and the Blogosphere is a much more dynamic ad hoc social network (compared to both Enron and Twitter), where groups and their memberships can change rapidly in a matter of days. This behavior is well reflected in the experiments described above.

Note also that the interaction probability method shows a behavior which is less extreme. This is due to the normalization by N^2 which ranged from 100^2 (Enron) to $2,700^2$ (Twitter) to $13,000^2$ (Blogs). Thus, the interaction probability based measures are heavily influenced by N, and work well for computing the evolution of a particular network. For comparing the evolution of different networks, it is perhaps more useful to first compare the evolution of the network with some null evolution such as the typical distance between random clusterings of the observed size distributions in C_1 and C_2. This way, one can normalize the observed distance by the distance for a random evolution which would be more appropriate for comparing between social networks. Nevertheless, these measures are appropriate, even without such normalization when it comes to comparing the performances of different algorithms on the same data set (with say a ground truth clustering).

IV. Conclusions

Measuring the distance between partitions is an important problem, which has received much attention. However, modern online data sets present examples where clusters or communities are necessarily overlapping and new methods for comparing such clusterings should be developed. We have made a step in this direction by presenting three novel definitions of the distance measure between collections of potentially overlapping sets, and we also gave algorithms for computing those distances. These algorithms are efficient (low order polynomial).

We tested our algorithms on diverse data sets, composed from social coalitions in the Blogosphere, Twitter and the Enron email network. Our results give the expected conclusions. Future work includes further analysis and improvement of proposed distance measures as well as potential construction of new measures suitable for structures in social networks and other domains. For social networks, where the generative interaction probability model is reasonable, the interaction probability measure is appropriate. The interaction probability model is based on the hypothesis that members of the same cluster are more likely to interact than members of different clusters. This is appropriate to any domain where clusters are defined in such a way (for example clustering books into topics based on user interaction data, or social networks). The best match and the K-center methods are more generally applicable to arbitrary collections of overlapping sets.

References

[1] J. Baumes, M. Goldberg, M. Hayvanovych, M. Magdon-Ismail, W. Wallace, and M. Zaki. Finding hidden group structure in a stream of communications. *Intelligence and Security Informatics (ISI)*, 2006.

[2] J. Baumes, M. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. *Intelligence and Security Informatics (ISI)*, pages 27–36, 2005.

[3] L. Denoeud and A. Guenoche. Comparison of distance indices between partitions. *Data Science and Classification*, pages 21–28, 2006.

[4] U. Feige. A threshold of ln(n) for approximating set cover. *Journal of the ACM (JACM)*, pages 634–652, 1998.

[5] S. Gregory. A fast algorithm to find overlapping communities in networks. *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2008.

[6] J. Kelly and B. Etling. Mapping iran's online public: Politics and culture in the persian blogosphere. *Berkman Center for Internet and Society at Harvard Law School*, 2008.

[7] A. Lancichinetti, S. Fortunato, and J. Kertesz. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 2009.

[8] M. Meila and A. Patrikainen. Comparing clusterings. *UW Technical Report, Statistics*, 2006.

[9] J. Munkers. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, pages 32–38, March 1957.

[10] T. Nepusz, A. Petrczi, L. Ngyessy, and F. Bazs. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 77, 2008.

[11] M. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci.*, 2005.

[12] G. Palla, I. Dernyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 2005.

[13] D. Zhou, J. Li, and H. Zha. A new mallows distance based metric for comparing clusterings. *ACM International Conference*, pages 1028 – 1035, 2005.