Agglomerative Genetic Algorithm for Clustering in Social Networks

Marek Lipczak
Faculty of Computer Science
Dalhousie University
Halifax, Canada, B3H 1W5
lipczak@cs.dal.ca

Evangelos Milios
Faculty of Computer Science
Dalhousie University
Halifax, Canada, B3H 1W5
eem@cs.dal.ca

ABSTRACT

Size and complexity of data repositories collaboratively created by Web users generate a need for new processing approaches. In this paper, we study the problem of detection of fine-grained communities of users in social networks, which can be defined as clustering with a large number of clusters. The practical size of social networks makes the traditional evolutionary based clustering approaches, which represent the entire clustering solution as one individual, hard to apply. We propose an Agglomerative Clustering Genetic Algorithm (ACGA): a population of clusters evolves from the initial state in which each cluster represents one user to a high quality clustering solution. Each step of the evolutionary process is performed locally, engaging only a small part of the social network limited to two clusters and their direct neighborhood. This makes the algorithm practically useful independently of the size of the network. Evaluation on two social network models indicates that ACGA is potentially able to detect communities with accuracy comparable or better than two typical centralized clustering algorithms even though ACGA works under much stricter conditions.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*

General Terms

Algorithms

Keywords

genetic algorithms, graph clustering, community detection, social networks

1. INTRODUCTION

Social network services allow users to create virtual personalities, represented by user profiles, and establish social connections. The user profile can gather personal information about the user and user's interests (e.g., *Facebook*[1]), about webpages the user bookmarked (e.g., *BibSonomy*[2]) or any other information that matches the type of social networking system. The social connections can be represented by explicit friendship links, or they can be extracted from various forms of people interaction (e.g., comments in blogging service, or citations in a repository of research papers). The social services allow users to access the information about related people by lists of friends or peers. Such a method seems awkward as humans are used to thinking about their social relations in terms of the communities they belong to. The information about community membership can be a valuable extension of social network services, making relations more natural than the basic friendship lists that are currently available. Despite the fact that community detection can be seen as a type of clustering problem, a well defined area of machine learning, the complexity of information gathered in social networks and their vast size forces us to redesign the traditional approach. Traditional methods, which assume that the entire dataset can be processed at once, are no longer practical. Instead, we propose to move toward progressive rearrangement of clusters in small subsets of the social network. In this paper we show how the genetic algorithm concept can be used to design an agglomerative clustering algorithm that is able to detect communities by processing at each step information related to users of two limited clusters.

Agglomerative clustering is an example of a hierarchical clustering approach. The algorithm result is a hierarchy of clusters; however, it can be easily modified to produce a flat partition of instances. The algorithm starts with a set of n clusters, each cluster represents one item from the dataset to be clustered. At each step two clusters are joined based on a given join criterion. The most frequently used criterion is the distance between two clusters that are to be joined. The algorithm needs also a stopping criterion to prevent the agglomeration process from continuing up to the point where all instances belong to one cluster. The concept of agglomerative clustering can be viewed as a greedy search method in which at each step we look for the currently optimal arrangement of clusters given the condition that two clusters must be joined. Let us assume that we look for the two clusters with a minimal average distance between members of both clusters. To find the currently optimal join the algorithm examines all possible pairs of clusters. Such an

[1]http://facebook.com
[2]http://bibsonomy.org

approach is impractical for the community detection problem because of the size of social networks and the number of expected communities. We could easily think of an extension of agglomerative clustering in which the algorithm examines a chosen pair of clusters and makes a decision if they should or should not be joined. The extension limits the computational and operational cost of the processing step; however, the local character of the decision can easily lead the algorithm to local optimum.

The objective of our work is to implement the ideas of such extended, local decision based, agglomerative clustering in the form of a genetic algorithm. The presented Agglomerative Clustering Genetic Algorithm (ACGA) at each step locally considers two clusters and based on the provided fitness function tries to join them or rearrange them such that the newly created cluster(s) improve the global clustering result. Both representation size and computational cost of one processing step is limited.

1.1 Communities in Social Networks

There is a large number of community definitions present in the literature, for example [4, 13, 16]. All of them carry the same notion of the subset of a graph (here understood as a graph of relations between users of a social network, which we refer to as *friendship links*) in which nodes (users) are densely connected, having at the same time low connectivity with the rest of the graph. Depending on the application, this definition should be refined by taking into consideration the required granularity of a community partition. For example, given a group of researchers profiled by their publications and connected by co-authorship or citation-based links we can think of different kinds of communities. We can partition the researchers based on their field, looking for few large communities (e.g., "theory", "databases" and "machine learning") [4]. However, we might also be interested in detecting the large number of communities that gather small groups of researchers that actually cooperate and frequently exchange ideas. In our work we focus on the latter notion of community, referring to them as *fine-grained* communities. This definition of a problem makes most of the traditional clustering techniques inapplicable as they assume that the number of clusters k is small and independent of the number of clustered items. This allows us to neglect the impact of k in determining the computational complexity of an algorithm. For the fine-grained community problem we should rather assume that the size of a community is small and limited, and the number of communities grows linearly with the number of users. The limited community size assumption is based on sociological and neurological observations of humans made by Dunbar [2]. Dunbar estimated the average size of a social group, in which humans are able to maintain stable group relations, to be 150, which is now referred to in the literature as the *Dunbar number*.

1.2 Community detection assumptions

In this section we enumerate all assumptions we make to transform a vague idea of community detection in social networks into a well defined clustering problem.

We assume that, because of the profile complexity, there is no possibility of maintaining a multidimensional space in which the position of the profile or a central point of a community can be determined. We shall instead rely on a similarity function able to calculate similarity scores between pairs of users. Such functions are already used in number of social services mostly to recommend products chosen by *"similar users"*. Again, it does not seem practical to calculate the similarity score between all pairs of users. We should limit ourselves to similarity scores calculated between users connected by a friendship link. These two assumptions define the dataset as a weighted graph were nodes represent users and edges are friendship links weighted by the profile similarity score. In this work we assume that both friendship graph and similarity scores are given.

Finally, when looking for *fine-grained* communities we should be aware that a user is likely to belong to more than one community. Because such overlapping clusters greatly increase the complexity of an already complicated problem, we decided to leave it out of scope of this paper.

1.3 Clustering as optimization problem

As defined in Section 1.2 we view community detection as a graph clustering problem. There are various measures of graph based clustering quality presented in the literature. Each of them can potentially be used as a fitness function that guides a genetic algorithm towards the optimal partition of clusters. We decided to focus on three cluster quality scores. The choice was made based on the expected behavior in boundary cases of the agglomeration process and an intuitive motivation of the score.

Two previously proposed optimization functions, *normalized cut* and *modularity* try to capture the intuition that a good community partition should minimize the number of links between clusters while keeping a high density of connections between community members. The *normalized cut*, proposed by Shi and Malik [16], assumes that a graph $G(V, E)$, where V is the set of all nodes and E is the set of all edges between these nodes, is to be divided into two disjoint sets A and B, where $B = V - A$. The score represents the fraction of all connections *cut* between A and B with respect to the number of connections involving nodes in A and B separately, Eq. 1. The latter is called *association* and is meant to balance the size of clusters.

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{1}$$

where $cut(A,B) = \sum_{i \in A, j \in B} w(i,j)$,
$assoc(A,V) = \sum_{i \in A, v \in V} w(i,v)$,
$w(i,j)$ is the weight of the edge between nodes i and j. In our application we assume that $w(i,j)$ is equal to a similarity score if i and j are connected by a friendship link or zero otherwise.

Assuming that B is the complement of A, the formula can be used to calculate the *normalized cut* for a single cluster C_k. Making this assumption we can transform Eq. 1 into:

$$Ncut(C_k, V - C_k) =$$

$$= \frac{cut(C_k, V - C_k) * assoc(V,V)}{assoc(C_k,V) * (assoc(V,V) - assoc(C_k,V))} \tag{2}$$

Given the function of a *normalized cut* for a single cluster, we define the *normalized cut* score for a complete k cluster partition as a weighted average of individual cluster scores, Eq. 3. Intuitively, large clusters should have larger impact on the score.

$$Ncut = \frac{\sum_k |C_k| * Ncut(C_k, V - C_k)}{|V|} \quad (3)$$

Using a similar rationale Newman and Girvan [13] proposed a *modularity* score. The score awards partitions that keep a high fraction of connections inside communities. *Modularity* was formulated based on matrix operations [13]; however it is possible to represent it in terms of cut and association scores, Eq. 4. Given this representation, the total value of *modularity* is a sum of scores calculated for individual clusters, Eq. 5.

$$Q(C_k) = \frac{assoc(C_k, C_k)}{assoc(V, V)} - (\frac{assoc(C_k, V)}{assoc(V, V)})^2 \quad (4)$$

$$Q = \sum_k Q(C_k) \quad (5)$$

This representation allows us to reveal the relation between *normalized cut* and *modularity* calculated for an individual cluster (transformation is based on the fact that $assoc(C_k, V) = assoc(C_k, C_k) + cut(C_k, V - C_k)$).

$$Q(C_k) = \frac{cut(C_k, V - C_k)}{assoc(V, V)}(\frac{1}{Ncut(C_k, V - C_k)} - 1) \quad (6)$$

Essentially *modularity* is a weighted inverse of the *normalized cut*. However, it is weighted by cluster's *cut* instead of cluster's size. Clusters with low *cut* value has their modularity score decreased which may be an additional factor that balances the size of clusters.

The third score that can be used as an optimization function is *silhouette width*, proposed by Rousseeuw [15]. The score is based on the intuition that a cluster should gather similar elements. The score is calculated for each individual item (node) i, Eq. 7. The total *silhouette width* is an average over all nodes in the graph.

$$S(i) = \frac{a'(i) - b'(i)}{max(a'(i), b'(i))} \quad (7)$$

where $a'(i) = avg_{j \in A}(w(i, j)), i \in A$
$b'(i) = max_C d'(i, C)$
$d'(i, C) = avg_{j \in C}(w(i, j)), i \notin C$

In his original work, Rousseeuw calculated the *silhouette width* based on dissimilarity matrix; however, the analogous formula for similarity matrix, Eq. 7, was also presented. In our work we use the similarity-based approach.

2. RELATED WORK

The problem of community detection in social networks using genetic algorithm was recently addressed by Pizzuti [14]. The algorithm uses a *locus-based* adjacency representation [6], where value j of gene on position i represents a link between instances i and j. All instances connected by a chain of links should be assigned to the same community. This representation does not require explicit information about the number of clusters; however, a decoding step is necessary to find all connected components. It can be done in $O(n)$ time. The algorithm optimizes a *community score* based on assumptions similar to *modularity*. A community detection GA based on *modularity* was presented by Tasgin and Bingol [17].

Community detection can be addressed by a broader class of clustering algorithms – network clustering. Feng et al. [3] proposed a genetic algorithm for network clustering. The algorithm operates on a weighted graph that represents similarities between network users. The authors modified the *modularity* score to capture hubs connected to many communities. According to the authors such hubs should be placed in distinct communities. Graph based clustering was also an object of multi-objective genetic algorithms. Uyar and Ogunducu [18] proposed a multi-objective GA approach extended by a heuristic post-processing step that limits the number of clusters created by the crossover operator. Two fitness functions are *min-max cut* and *silhouette width*.

A variety of multi-objective clustering genetic algorithms was proposed by Handl and Knowles [6]. The most recent approach, MOCK uses two clustering criteria – *compactness* and *connectedness* as fitness functions. The number of clusters does not have to be explicitly defined thanks to the *locus-based* representation. The fitness functions proposed by the authors operate in a multidimensional attribute space. A possible extension to graph-based datasets can be made using a semi-supervised modification of MOCK [7]. Unfortunately the high computational cost of MOCK may make it inapplicable to realistic social networks. Despite some recent improvements in the computational cost of the cluster number determination step [12] there are still issues of processing the initialization step based on a minimal spanning tree and the *locus-based* representation decoding.

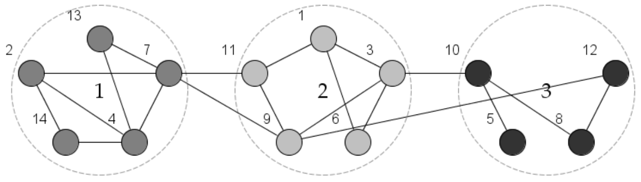
The above examples indicate that a GA may be used to optimize one or more clustering quality scores to produce high quality clusters. Each of the presented approaches assumes that the complete dataset can be represented as a single individual and processed at once. The main objective of our work was to reduce the size of an individual and the computation cost of a single fitness score to make the algorithm feasible given the real sizes of social networks.

3. AGGLOMERATIVE CLUSTERING GENETIC ALGORITHM

In standard approaches to applying GAs to clustering problems each individual denotes a complete solution. The solutions compete and exchange genetic material evolving toward a well-fitted individual that represents the final partition. In our *Agglomerative Clustering Genetic Algorithm* (ACGA) each individual represents one cluster. The crossover operator allows two individuals to exchange genetic material of two clusters to locally improve the value of fitness function. The number of individuals is equal to the current number of clusters and the complete solution is represented by the whole population. Given the assumption that a cluster size is small and limited (fine-grained communities) this modification allows the algorithm to calculate the value of the fitness function engaging only a small part of the graph at a time, hence making the processing feasible independently of the size of the social graph. This section describes modifications of genetic operators required by ACGA.

Representation.

Clustering is an optimization problem that can be mapped to a variety of GA representations. The straight-forward representation, *string-of-group* [9] encoding represents each individual as a string of n integers (genes), where n is the



(a) An example of social network with 14 users (nodes) grouped into three communities.

Individual "A"

id	1	2	3	4	5	6	7	8	9	10	11	12	13	14
value	2	1	2	1	3	2	1	3	2	3	2	3	1	1

(b) Straight-forward *string-of-group* encoding, where each individual represents a complete clustering solution.

Individual "A"

id	2	4	7	13	14
value	1	1	1	1	1

Individual "B"

id	1	3	6	9	11
value	2	2	2	2	2

Individual "C"

id	5	8	10	12
value	3	3	3	3

(c) Localized encoding, each individual represents one cluster, the complete solution is represented by the whole population. The inactive genes are not included in the representation of individual, which is thus of variable length.

Figure 1: Clustering solution and its representation.

number of items (i.e., users in community detection problem) in the dataset. The genes take values from 1 to k, where k is the expected number of clusters (Fig. 1(b)). This fixed-length, fixed-order representation can be used in a broad range of genetic algorithms. When the values of n and k are high this representation becomes impractical, and not only because of the large length of individual encoding. The huge search space(n^k possible distinct individuals) creates the need for an initialization function. In addition, obtaining a value for the fitness function for the complete individual is likely to be time consuming. Analogous issues can be found in the *locus-based* representation, presented in Section 2.

These shortcomings lead us to a simplified representation that can be used in the agglomerative clustering approach. In ACGA each individual represents one cluster. Using the *string-of-group* encoding we can think of an individual as a string of n genes, where n is the number of items collected in the cluster represented by the individual (Fig. 1(c)).

Selection operator.

The algorithm is based on a two-step steady-state selection operator, with an elitist replacement strategy ($\mu + 2$). The introduced localized representation requires significant modifications on the selection operator. Selection of the two individuals to be recombined cannot be based on the fitness value, as each individual represents only a part of clustering solution and there is no notion of fitter individuals among the population. ACGA chooses parents based on the potential improvement of their fitness rather than the fitness prior to recombination. The choice of the first parent is done by a random activation of a node, which represents a user in a social network. The active node checks the similarity of neighbors (user's friends) that are not members of its cluster. A tournament selection scheme is used to pick a similar user which defines the cluster individual that becomes the second mate. Given the pair of parents, the crossover operator is used to create the pair of children. The pair of parents is replaced by the pair of children only if the their fitness is higher than the fitness of parents. The comparison of fitnesses of parents and children is legitimate as they both represent identical subset of the graph. It is important to notice that the definition of each clustering quality function presented in Section 1.3 allows us to calculate a local fitness value for any subset of clusters.

Involving nodes (users) in the selection process is not accidental. The information at the node level is easy to retrieve and process, which reduces the processing cost of the selection step. The proposed selection method makes the probability of choosing an individual proportional to the size of the cluster it represents. This is sensible as large clusters have more genetic material and are more likely to create interesting new partitions. The method of the second individual choice is meant to reduce the search space, ensuring that two clusters are tightly connected by at least one pair of users which are similar.

Crossover operator.

The variable length representation of individuals can be mapped to a fixed-length representation by adding null genes in place of genes present in the other parent. We make the standard assumption that each item has a unique id which defines the position of the gene representing it in a chromosome. The genes that are present in none of the parents can be omitted as they have no impact on the result of crossover. After that, standard crossover operators can be applied (e.g., one-point crossover or uniform crossover). The informative genes taken from the base representation are, after the mapping, distributed uniformly in the chromosomes of parents. This is the reason why one-point crossover tends to equalize their number in children individuals (Fig. 2). We observe a similar situation for uniform crossover operator. As the decision about each gene is made independently, the distribution of the number of informative genes in the two children individuals after uniform crossover follows the binomial distribution ($p = 0.5$, $n = |C_i + C_j|$). Again uniform crossover tends to produce chromosomes with similar numbers of informative genes (i.e., clusters of similar sizes). When mixing small and large clusters we should take their sizes into consideration, otherwise we will result in two clusters of similar size that is about half of the size of the large cluster. It makes joining clusters (i.e., creating a pair of children in which one has only null genes) less probable and blocks the agglomeration process when the number of informative genes in parent clusters increases. To represent the size imbalance between two parent clusters we can utilize the flexibility of uniform crossover. It can be biased by reducing the probability p of applying crossover to the informative genes of individual with the higher number of them. In experiments we use a parametrized version of uniform crossover. The impact of uniform distribution versus the biasing factor is defined by parameter p_{uni}, Eq. 8. When $p_{uni} = 0.5$ there is no bias.

$$p_{Size} = p_{uni} + (1 - 2p_{uni})\frac{|C_j|}{|C_i| + |C_j|},\ p_{uni} \in [0, 0.5] \quad (8)$$

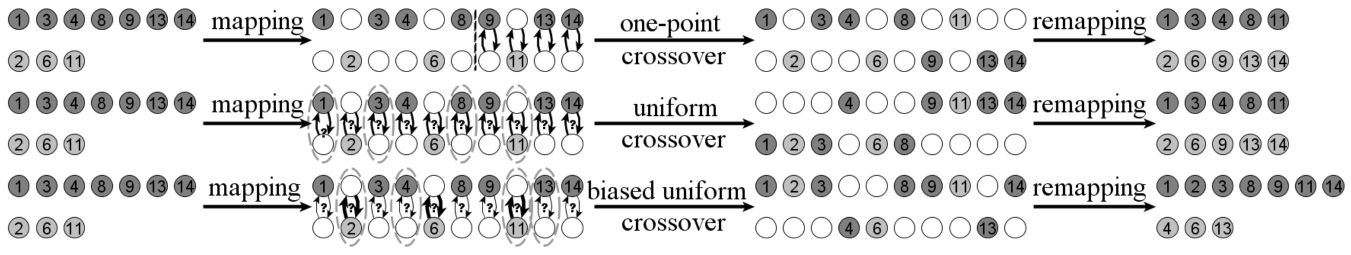


Figure 2: Three types of crossover on individuals mapped to fixed-length representation. Despite a different number of informative genes in parents, one-point and unbiased uniform crossover tend to produce children with equalized number of informative genes. Biased crossover makes informative genes from the "larger" parent less likely (and the "smaller" parent more likely) to be crossed-over, which keeps the cluster imbalance.

Fitness function.

Each of the three clustering quality scores presented in Section 1.3 were independently assessed as fitness functions. Single fitness value is calculated for a pair of children. We use the general formula for calculating the score for n clusters, but a set of clusters is limited to a pair of children. Even though fitness is calculated for the full graph of connections, each objective function can be calculated based on the local information only (i.e., friendship links of users gathered in both clusters). The full graph association $assoc(V, V)$ that appears in the formula of *normalized cut* and *modularity* is constant and can be estimated beforehand. *Normalized cut* and *modularity* are additive: by improving the score for a subset of clusters we always improve the total score. This is not true for *silhouette width*, as changing a cluster may cause a change of the cluster with the maximal similarity score in any of its neighbors.

4. EVALUATION

The standard method of clustering algorithm evaluation is to perform a comparison with baseline algorithms using datasets for which the expected clustering result is known. Most of the datasets introduced in the literature contain labels that partition the dataset into a small number of communities. This makes them applicable to the coarse-grained community detection algorithms. Labeling of fine-grained communities is more complicated and appears to be an interest of sociologists rather than computer scientists. The datasets created in sociological studies (e.g., Zachary Karate club proposed as a clustering dataset by Girvan and Newman [5]) are too small to capture the properties of the clustering algorithms. We therefore decided to process the evaluation based on synthetic data generated out of two social network models.

4.1 Social network models

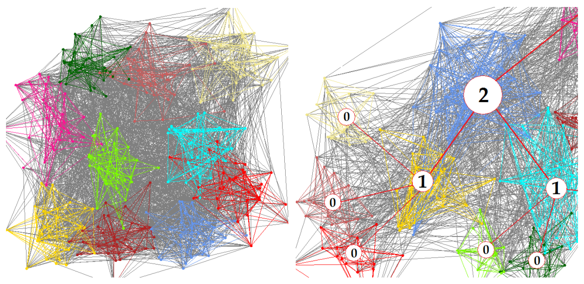
The plain model is an adaptation of a widely-used model proposed by Girvan and Newman [5]. Given N instances we distribute them uniformly to k clusters. For each pair of users we randomly decide if they should be connected. The probability of establishing a friendship link $P(link)$ is higher if two users are members of the same community ($P(link_{in}) > P(link_{out})$). The model was proposed for binary graphs. To add similarity weights we uniformly distribute the centroids of clusters in d dimensional, normalized space. Points p_i, which represent instances (users), are distributed around centroids using the Gaussian distribution. The similarity value is calculated based on Euclidean distance, Eq. 9. We set the parameters of the model to keep the average number z_{in} of the inner links of an instance lower than the number of outer links if $k > 2$ (Table 1(a)). As the outer links are distributed over $k - 1$ clusters, the correct partition can be determined. In addition, to make the hardness of clustering less dependent on increasing number of clusters the similarity factor is used. Similarity has impact on link generation, making similar instances more likely to be connected, Eq. 10.

$$sim(i, j) = 1 - \frac{|p_i, p_j|}{\sqrt{d}} \tag{9}$$

$$P(link) = P(link_{in|out})\beta_P{}^{1-sim(i,j)} \tag{10}$$

The hierarchical model is a modification of *community guided attachment* (CGA) model proposed by Leskovec et al. [11]. To model some characteristics observed in real social networks the authors built a recursive model creating a hierarchy of communities. To introduce the rationale behind the model we will use an example of a community tree found in large companies or institutions. The root of the tree is occupied by a large community of executives, going down through levels of departments we finally reach small task-oriented communities of workers. We can expect that most of the social connections take place between members of the same community. The inter-community connections are less likely if the distance in the community hierarchy is large. As the transport of resources and information in large institutions is vertical, two communities in different departments can work on similar problems not knowing about each other. The input to the CGA model is a balanced tree of communities with height H and number of branches b. We decided to use the version of the CGA model in which communities are present at each level of the tree. The probability of establishing a friendship link between two users decreases exponentially with the distance between their communities in the hierarchy, Eq. 11. The similarity weights are calculated as in the plain model, but here they do not influence the probability of creating links. The centroids of clusters are generated within a limited range of their parents ($range_{min}$, $range_{max}$). Finally the number of instances in a cluster and their variance grows with the level of the cluster l (for leaves $l_{leaf} = 0$). Example parameters are shown in Table 1(b).

$$P(link) = P_{base}\beta_H{}^{treeDist(i,j)} \tag{11}$$



(a) Plain model. (b) Hierarchical model (with hierarchy levels).

Figure 3: Datasets generated by models (friendship links within the same community colored).

Table 1: Values of parameters of the two social network models for two example test datasets each. The parameters are introduced in Section 4.1

(a) plain model

parameter	P2x500	P100x10
N	1000	1000
k	2	100
$P(link_{in})$	0.3	0.3
$P(link_{out})$	0.15	0.15
β_P	0.01	0.01
d	10	10
space Std	0.1	0.018
derived parameters (experimental)		
$avg\ z_{in}$	80	3.6
$avg\ z_{out}$	19	27.5

(b) hierarchical model

parameter	Hb3H5	Hb10H3
H	5	3
b	3	10
P_{base}	0.4	0.3
β_H	0.31	0.33
g	1.5	2.0
$\lvert C_k \rvert$	$16 * g^l$	$15 * g^l$
$range_{min}$	$0.1 * g^l$	$0.1 * g^l$
$range_{max}$	$0.2 * g^l$	$0.3 * g^l$
d	10	10
space Std	$0.1 * g^l$	$0.1 * g^l$
derived parameters (experimental)		
N	2511	1860
k	121	111
$avg\ z_{in}$	9	5.3
$avg\ z_{out}$	14	23.9

4.2 Baseline algorithms

The results of ACGA were compared to an agglomerative clustering algorithm which in each step merges the two closest clusters based on average distance. This approach, named *Unweighted Pair-Group Method* (UPGMA), is widely used in bioinformatics [1]. The stopping criterion can be the number of true clusters or the maximal value of a clustering score. In the latter approach, instead of stopping the agglomeration process when it reaches the correct number of clusters, we let it complete and then pick the agglomeration step that has the best value for one of the scores. This approach was used to gain more information about the three fitness functions used in experiments. In addition, we present the results of a normalized cut optimization algorithm based on recursive bisection. This divisive clustering approach was proposed by Shi and Malik [16].

4.3 Methodology

The algorithms were evaluated based on labels created by the data generation process. The clustering results and true clusters were presented in a confusion matrix, which was used to calculate the *adjusted Rand Index* [10]. This metric examines all pairs of instances. It calculates the total number of pairs that belong to the same cluster, or to different clusters, at the same time both in true clusters and clustering results. The score is adjusted so that the value for perfectly clustered data is one.

We experimented with two other standard metrics used to evaluate the quality of clustering results: entropy and purity [8]. We found them not reliable when the number of result clusters is different from the number of true clusters.

5. EXPERIMENTAL RESULTS

Among a large number of features and parameters of genetic algorithms we decided to focus on fitness function and crossover operator. The fitness function, which tries to define the quality of a clustering result, plays the most important role in any evolutionary based clustering algorithm. The crossover operator is especially important in the *Agglomerative Clustering Genetic Algorithm* as other operators are limited by the specific representation of individuals and local character of processing. We tested three fitness functions and various settings of the biased uniform crossover operator. We measured the quality of clustering according to the *adjusted Rand Index*, and the number of result clusters with respect to that of real clusters in two experiments. The first experiment was a sequence of runs on datasets generated by the plain model. Each dataset consisted of 1000 nodes. Starting with two clusters (`P2x500`) we gradually increased the number of clusters and decreased the standard deviation of the distribution of points to observe how the number of clusters and their size influence the performance of ACGA. The plots in Fig. 4 show the results for datasets with 2, 4, 8, 10, 15, 20, 33, 50, 66 and 100 clusters. In the second experiment we ran the algorithm on datasets generated by the hierarchical model with various settings. The aim of this experiment was to confirm the quality of the results obtained for the plain model on more complex data. Because of space limitations we present only two example results for a deep hierarchy of five levels with three branches (`Hb3H5`) and a broad hierarchy of three levels with ten branches (`Hb10H3`) (Table 2). The specific parameters for the datasets are presented in Table 1.

The presented results are an average over runs on 10 datasets generated from a model based on the same parameters. ACGA was run for 500 epochs.

5.1 Fitness functions

The results of ACGA based on three different fitness functions are very consistent over different datasets generated by the plain model. The strong relation between *normalized cut* and *modularity* revealed in Section 1.3 was confirmed in the experiments. For any combination of crossover operator bias and plain model dataset, ACGA based on *normalized cut* and *modularity* give nearly identical results. For this reason results for the plain dataset are presented for only one of them (*modularity*) (Fig. 4). The tests on the hierarchical model datasets showed, however, a slight difference between behavior of ACGA that optimizes *normalized cut* or *modularity*. *Modularity* is more likely to join two clusters that are close in the hierarchy tree which leads to over-agglomeration. Possibly, as only a small part of graph is processed, *cut* factor, which should balance the clusters in *modularity* score, does not work properly.

In all tests we could observe that, when optimizing *silhouette width* the algorithm is not able to pass the early stage of evolution in which small clusters (usually four or five densely connected nodes) are formed. Both for for plain and hierarchical model datasets, the value of the *silhouette width* for such arrangement is often higher than the value of the correct solution. Although *silhouette width* cannot be used to drive an agglomeration process, it can be used to estimate the correct number of clusters when the agglomeration process is done by an external algorithm like UPGMA (see results of `UPGMA S`).

5.2 Crossover operator

As expected, the results indicate that ACGA with a uniform crossover operator is not able to create large clusters. At some point, in order to extend clusters to a larger size, the algorithm must merge small groups into their larger-sized neighbors. Biasing the crossover operator allows the agglomeration process to continue. Removing the random factor ($p_{uni} = 0$) makes small clusters very likely to be merged into large neighbors. It allows the algorithm to find high quality partitions consisting of a small number of large clusters. As the number of clusters grows, the tendency of biasing approaches to over-agglomerate becomes more visible. This is caused by the characteristics of fitness functions (*normalized cut* and *modularity* scores) which prefer small number of large clusters. Using UPGMA as an optimization algorithm for these clustering quality scores reveals that *normalized cut* (`UPGMA N`) and *modularity* (`UPGMA Q`) underestimate the number of clusters if the true number of clusters is high. This is probably the reason why strongly biased crossover causes ACGA to over-agglomerate.

The results of ACGA for plain model datasets indicate that finding a good solution requires the impact of bias and random factor in the uniform crossover to be balanced in order to match the expected granularity of clusters. The choice of bias parameter is dataset dependent, which is an expected disadvantage of any parametrized algorithm. For datasets used in the experiments ACGA was able to reach best results when the impact of both factors was similar (e.g., $p_{uni} = 0.2$, $p_{uni} = 0.3$). For these settings the algorithm showed the best flexibility in finding clusters of different size. It was especially important while tested on hierarchical model datasets, where the algorithm was able to detect clusters on each level of the hierarchy (e.g., cluster size varied from 16 to 81 for `Hb3H5` dataset).

6. CONCLUSIONS AND FUTURE WORK

Looking for communities in social networks we cannot assume that the complete dataset can be processed at once. This constraint makes most of the clustering algorithms, including GA solutions, impractical for this task. The concept of genetic algorithm can be, however, used to built an agglomerative clustering algorithm. In the proposed *Agglomerative Clustering Genetic Algorithm* each individual represents one cluster, instead of the whole clustering solution. This allows a pair of individuals to recombine or join the genetic material that is a small subset of the network. The evaluation of newly created individuals can be done based on two clusters and their direct neighborhood. If ACGA is used to look for clusters of limited size, for example fine-grained communities in social networks, each of its steps requires limited resources irrespective of the dataset size.

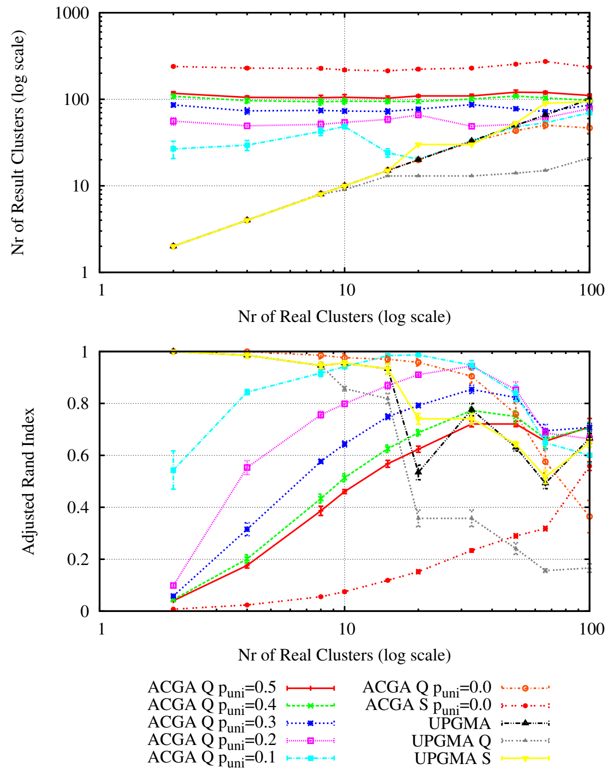


Figure 4: Results of ACGA and UPGMA algorithms for a sequence of plain model datasets with increasing number of clusters $k \in [2, 100]$. As the results for *normalized cut* and *modularity* were nearly identical, we decided to show only the latter for clarity. The plots show the results of six crossover biases for modularity score (ACGA Q ...), from the weakest ($p_{uni} = 0.5$) to the strongest ($p_{uni} = 0$); example result for *silhouette width* (ACGA S $p_{uni} = 0$); and results of UPGMA where the result was chosen by the true number of clusters (UPGMA) or the maximal value of *silhouette width* (UPGMA S) or *modularity* (UPGMA Q).

The evaluation on datasets generated by two social network models demonstrated that ACGA was generally able to match and often outperform the UPGMA algorithm. It was possible even though there is no fair competition between these two approaches. ACGA is based on stricter assumptions – unlike UPGMA, it must make decisions locally without the full information about the dataset, including the true number of clusters. Comparison of two baseline algorithms UPGMA and *normalized cut* heuristic indicates that the agglomerative approach is preferable for fine-grained community detection. Conversely, the *normalized cut* heuristic is able to find a perfect partition when the number of clusters is low and at each step they can be divided into two parts. Otherwise, errors made at the beginning of the recursion are propagated and subdivisions lose sense. However, agglomeration puts hard demands on an algorithm, which must process beyond the early phase of agglomeration when items are likely to form small cliques, and later it must prevent the process from over-agglomeration. These expectations seem to be too high for any fitness function. In ACGA the

Table 2: Results of ACGA, UPGMA and *normalized cut* heuristic for two hierarchical model datasets. The table shows the results of ACGA with six crossover biases for *normalized cut* (ACGA N ...) and modularity score (ACGA Q ...); two boundary crossover biases for *silhouette width* (ACGA S ...); and results of UPGMA algorithm where the result was chosen by the true number of clusters (UPGMA) or the optimal value of any of the three clustering quality scores (UPGMA N, UPGMA Q, UPGMA S).

	Hb3H5		Hb10H3	
Algorithm	adjRandInd	k	adjRandInd	k
ACGA N $p_{uni} = 0.5$	0.77	302.4	0.62	226.0
ACGA N $p_{uni} = 0.4$	0.81	262.7	0.66	189.3
ACGA N $p_{uni} = 0.3$	0.86	175.3	**0.69**	140.9
ACGA N $p_{uni} = 0.2$	**0.88**	138.8	0.65	112.3
ACGA N $p_{uni} = 0.1$	0.82	106.2	0.34	70.7
ACGA N $p_{uni} = 0.0$	0.76	93.6	0.18	25.5
ACGA Q $p_{uni} = 0.5$	0.72	312.1	0.55	224.3
ACGA Q $p_{uni} = 0.4$	0.76	267.5	0.56	190.2
ACGA Q $p_{uni} = 0.3$	0.79	182.0	0.59	135.3
ACGA Q $p_{uni} = 0.2$	0.81	129.3	0.57	100.1
ACGA Q $p_{uni} = 0.1$	0.73	91.7	0.36	69.9
ACGA Q $p_{uni} = 0.0$	0.73	87.8	0.17	20.1
ACGA S $p_{uni} = 0.5$	0.28	539.7	0.23	500.9
ACGA S $p_{uni} = 0.0$	0.36	453.4	0.28	460.3
UPGMA	**0.88**	121.0	0.53	111.0
UPGMA N	0.01	2.0	0.00	2.0
UPGMA Q	0.13	9.0	0.14	10.0
UPGMA S	0.87	117.0	0.58	135.0
nCut heuristic	0.38	121.0	0.17	111.0

preferred granularity is indirectly chosen by crossover bias.

An important advantage of the local processing in ACGA is the possibility of distribution of its computation. One of our future work goals is the distribution of the algorithm over the machines of users of a social network. This can assure that the algorithm is not only able to process large networks but it is also scalable as the processing power grows linearly with the number of clustered items.

The code, datasets and Web version of the ACGA visualization are available at `www.cs.dal.ca/~lipczak/acga.html`

Acknowledgments

The research was funded by the Natural Sciences and Engineering Research Council of Canada, the MITACS NCE, and Genieknows.com. We would like to thank Dirk Arnold and Malcolm Heywood from Dalhousie University for their comments and valuable input.

7. REFERENCES

[1] Robert Busa-Fekete, Andras Kocsor, and Csaba Bagyinka. A multi-stack based phylogenetic tree building method. *Bioinformatics Research and Applications*, pages 49–60, 2007.

[2] R. I. M. Dunbar. Coevolution of neocortical size, group size and language in humans. *Behavioral and Brain Sciences*, 16(4):681–735, 1993.

[3] Zhidan Feng, Xiaowei Xu, Nurcan Yuruk, and Thomas Schweiger. A novel similarity-based modularity function for graph partitioning. *Data Warehousing and Knowledge Discovery*, pages 385–396, 2007.

[4] Rong Ge, Martin Ester, Byron J. Gao, Zengjian Hu, Binay Bhattacharya, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. *ACM Trans. Knowl. Discov. Data*, 2(2):1–35, 2008.

[5] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[6] J. Handl and J. Knowles. An evolutionary approach to multiobjective clustering. *Evolutionary Computation, IEEE Transactions on*, 11(1):56–76, Feb. 2007.

[7] Julia Handl and Joshua Knowles. On semi-supervised clustering via multiobjective optimization. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1465–1472, New York, NY, USA, 2006. ACM.

[8] Ji He, Ah-Hwee Tan, Chew-Lim Tan, and Sam-Yuan Sung. *On Quantitative Evaluation of Clustering Systems*. Kluwer Academic Publishers, 2003.

[9] Yi Hong, Sam Kwong, Hui Xiong, and Qingsheng Ren. Genetic-guided semi-supervised clustering algorithm with instance-level constraints. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1381–1388, New York, NY, USA, 2008. ACM.

[10] Lawrence Hubert and Phipps Arabie. Comparing partitions. *J. of Classification*, 2(1):193–218, 1985.

[11] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2, 2007.

[12] Nobukazu Matake, Tomoyuki Hiroyasu, Mitsunori Miki, and Tomoharu Senda. Multiobjective clustering with automatic k-determination for large-scale data. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 861–868, New York, NY, USA, 2007. ACM.

[13] M E Newman and M Girvan. Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 69(2):026113.1–15, Feb 2004.

[14] Clara Pizzuti. Ga-net: A genetic algorithm for community detection in social networks. In *PPSN*, volume 5199 of *Lecture Notes in Computer Science*, pages 1081–1090. Springer, 2008.

[15] Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[16] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[17] Mursel Tasgin and Haluk Bingol. Community detection in complex networks using genetic algorithm. In *ECCS '06: Proc. of the European Conference on Complex Systems*, Apr 2006.

[18] A. Sima Uyar and Sule Gunduz Oguducu. A new graph-based evolutionary approach to sequence clustering. In *ICMLA '05: Proceedings of the Fourth International Conference on Machine Learning and Applications*, pages 273–278. IEEE, 2005.