A New Graph-Based Evolutionary Approach to Sequence Clustering

A. Şima Uyar
Department of Computer Engineering
Istanbul Technical University
Maslak, Istanbul TR34469
Turkey
etaner@cs.itu.edu.tr

Şule Gündüz Öğüdücü
Department of Computer Engineering
Istanbul Technical University
Maslak, Istanbul TR34469
Turkey
gunduz@cs.itu.edu.tr

Abstract

Clustering methods provide users with methods to summarize and organize the huge amount of data in order to help them find what they are looking for. However, one of the drawbacks of clustering algorithms is that the result may vary greatly when using different clustering criteria. In this paper, we present a new clustering algorithm based on graph partitioning approach that only considers the pairwise similarities. The algorithm makes no assumptions about the size or the number of clusters. Besides this, the algorithm can make use of multiple clustering criteria functions. We will present experimental results on a synthetic data set and a real world web log data. Our experiments indicate that our clustering algorithm can efficiently cluster data items without any constraints on the number of clusters.

1. Introduction

The problem of clustering unlabelled data objects into groups of similar objects is one of the important unsupervised problems in data mining and has been very well studied [11, 16]. Different clustering techniques in data mining are described and a comprehensive review of them is provided in two recent surveys [2, 15]. The clustering problem appears in two different forms according to the representation of data. Most of the clustering algorithms focus on clustering data items in metric spaces. However, in a large number of applications where the data items are in sequential form, the only information available is a similarity/dissimilarity measure between every pair of data items. This kind of data occurs in many application domains, such as biostatistics, medicine, telecommunications, user interface studies, market basket data, and World Wide Web (WWW) page request monitoring. Understanding the structure of such data still remains a challenge. For this reason, sequence clustering has become increasingly important. A representation of sequence data is a weighted, undirected graph where each sequence in the data set becomes a vertex in the graph. The pairwise similarities of sequences form the edges of the graph. The sequence clustering problem can be mapped then to graph partitioning [1] problem.

In this paper, we present a new approach for partitioning a weighted undirected graph. Graph partitioning is an NP hard problem [1], because of the combinatoric nature of the problem. For this reason, evolutionary algorithms (EA) are good solutions approach to optimize graph partitioning problems. To optimize artificial systems, EAs [10, 18] model the evolutionary process and the principles of Mendelian genetics found in nature. They are considered to be among the most powerful heuristic search and optimization methods and are commonly used to attack NP-hard problems. In this paper our objective is to develop a clustering algorithm with the following properties:

1. The input data set consists of pairwise similarities/dissimilarities of data items in order to be able to cluster sequential data.
2. We make no assumptions about the number of clusters. The algorithm will determine this value based on the data.
3. The algorithm can use multiple criteria functions which will result in robust clustering solutions.

As far as we know, existing clustering algorithms that respect those three properties are hard to find. Despite the extensive literature on clustering data items in metric space, there are fewer pairwise clustering algorithms [3, 20]. However, those algorithms do not automatically determine the number of clusters. Therefore, we concentrate in this study on a graph clustering method based on a multiobjective evolutionary algorithm that effectively identifies clusters. Our

[1] In this paper we will use "graph partitioning" and "graph clustering" interchangeably.

Proceedings of the Fourth International Conference on Machine Learning and Applications (ICMLA'05)
0-7695-2495-8/05 $20.00 © 2005 IEEE

algorithm uses two objectives based on the graph connectedness and a cluster validity index for graph partitioning. This approach to graph partitioning is novel and unique. We experimentally evaluated our method by using both a synthetic data set and a real world data set obtained from WWW page requests of a Web site. These graphs are highly irregular or random, and vertex degrees vary dramatically. The preliminary experimental results show that our method is successful in determining the number of clusters and improve the quality of clusters. Equally important, these results are robust across graphs with different structures. However this is a work in progress and there seem to be quite a few improvements to be made to the approach to even obtain better results in a shorter time. The current promising results promote further study.

The rest of the paper is organized as follows. In Section 2, we examine related work. Section 3 briefly describes graph based clustering. Section 4 presents our proposed method for graph partitioning. Section 5 provides experimental results. Finally, in Section 6 we conclude and discuss future work.

2. Related Work

An important form of data considered in data mining is sequential data. This kind of data occurs in many applications domains, such as biostatistics, medicine, telecommunication, user interface studies, market basket data, and World Wide Web page request monitoring. However, most research on clustering algorithms focus on non-sequential domain. All these algorithms assume that the data set is in metric space. However, the structure of sequences makes it difficult to use metric space. Each sequence is composed of non-numerical symbols, and the length of a sequence can run up to a thousand or even beyond. Many sequence clustering algorithms precompute all pairwise sequence similarities. In this case, the sequence data can be represented by an undirected graph *G* whose vertices are sequences in the data set. An edge connecting two vertices in the graph has a weight equal to the similarity between these two sequences. Properties of a graph can then be used to cluster sequences by constructing a set of subgraphs from *G*. Thus, sequence clustering problem becomes graph partitioning problem.

Most sequence clustering algorithms refine clustering recursively. However, one of the most challenging problems in sequence clustering methods that use graph partitioning approaches is to put a single cutoff score that separates all sequence clusters. At any given point a set of sequence clusters or subgraphs such as $\{G_1, G_2, ..\}$ are given. The problem in such methods is, should a cluster, for example, say, G_2 be split further or not. Different cutoff values may result in different clustering solutions. Many clustering algorithms use graph properties to handle this problem. The clustering algorithms in [14, 17] are based on a purely graph theoretic approach. However, there remains another important issue. In the simplest MIN cut algorithm, a connected graph is partitioned into two subgraphs with the cut size minimized. This leads to a skewed cut, where a subgraph could be very small compared to the other subgraphs. Various constraints are introduced, such as ratio cut [4], normalized cut [21], and min-max cut [9], etc. to remedy this problem. A commonly used graph clustering program is Cluto [5], where the program requires the number of clusters as input.

Evolutionary algorithms for clustering has been studied in literature. In [12] a steady-state genetic algorithm for clustering data items in metric space is given which is similar to k-means clustering algorithm. In [8] we proposed a new sequence clustering algorithm based on a hybrid evolutionary algorithm. However, both of these algorithms are unable to automatically determine the number of clusters, which is a major problem for clustering algorithms. A typical example for a evolutionary clustering algorithm is [13], which is based on k-means algorithm. This algorithm uses multiobjective optimization and determines the number of clusters. However, it expects data to be mapped into a geometric space and uses the metric distance between data points. For this reason, it is not appropriate for data sets where the data items are in sequential form. The crucial difference between our algorithm and these previous algorithms is that our algorithm clusters the sequential data using multiobjective optimization and does not require the number of parameters as an input. Consequently, as the experiments demonstrate, it is robust across graphs with different structures.

3. Graph Clustering Problem

Formally, graph partitioning problem is defined as follows: Given a graph $G = (V, E)$ where V is a set of vertices and E is a set of edges, partition the graph G into k disjoint subgraphs, such that the clustering criteria function is optimized. An edge connecting two vertices in the graph has a weight equal to the similarity between these vertices. We often identify a cluster C_i with the induced subgraph G_i.

4. The Evolutionary Graph Clustering Algorithm

For addressing the graph clustering problem with an evolutionary algorithm (EA), we used a standard steady-state EA with duplicate elimination and added a heuristic operator to exploit problem specific information and increase performance. The basic flow of the EA used is given in Algorithm 1.

Algorithm 1 Flow of the steady-state EA

```
1: randomly initialize population
2: while max no of fitness evaluations not reached do
3:    select parents
4:    create child through cross-over
5:    mutate child
6:    if child is not duplicate then
7:       apply heuristic disband to child
8:       child replaces worst in population
9:    else
10:      discard child
11:   end if
12: end while
```

In a steady-state EA, for each iteration of the algorithm, a new child is created through cross-over and mutation from two parents. In our approach, the child is further corrected using a heuristic operator. If the resulting child is the same as another individual which is currently in the population, the child is discarded. If a child is not a duplicate, its fitness is calculated. If the fitness of the child individual is better than or equal to the fitness of the worst individual in the current population, the child replaces the worst individual.

The main components of any EA are the method chosen to represent solution candidates, selection, cross-over and mutation techniques and the fitness evaluation method. These and the new heuristic operator will be explained in detail in the following subsections.

4.1. Representation of Solutions

The way solution candidates, which correspond to individuals in an EA, are represented is very critical to the performance of an EA. For the graph clustering problem, the most obvious choice is to have each gene represent a vertex and the value of the gene denote the cluster number the vertex is placed in. A sample individual for a clustering of 7 vertices into 3 clusters, can be seen below:

Individual 1: 1 2 2 1 3 1 3

This individual means that vertices (1,4,6) are in cluster 1, vertices (2,3) are in cluster 2 and vertices (5,7) are in cluster 3. Even though this representation is very straightforward, it has a limitation. For the graph clustering problem, a solution shows which vertices are clustered together and the actual cluster number is not relevant. Based on the selected representation method, the following sample individual represents a different solution candidate.

Individual 2: 2 3 3 2 1 2 1

However, for the graph clustering problem, this individual represents the same clustering solution as individual 1. This issue leads to two major problems which we address in this paper:

1. The search space increases artificially. As far as the EA is concerned, the search space size for this representation is c^n, where n is the number of vertices and c is the number of clusters. However, the number of different clustering solutions is less than the search space size determined by the representation.

2. The main purpose of cross-over should be to combine parts of the clustering solution in each parent to form a new child individual which is partially similar to its parents and is also different from them. Standard uniform cross-over becomes meaningless since the cluster numbers may have different meanings in the different parents and no meaningful information can be passed on from the parents to the child when they are combined.

For the solution of the first problem, we added a post-processing step to creating new individuals through initialization and also through cross-over and mutation. During this step, the individual is processed from left to right and the clusters are re-numbered in increasing order, starting from 1. As a result of this step, both sample individuals given above are converted to the same individual as given.

Individual 1&2: 1 2 2 1 3 1 3

However, this does not solve the second problem. A new cross-over operator is still needed. This issue will be explored in detail in the next subsection.

4.2. The Cross-Over Operator

A new heuristic uniform cross-over operator that aims at preserving clustering information from the parents is proposed. Assume that two sample parents are given as below.

Parent 1: 1 2 2 1 3 1 3	**Parent 2:** 1 1 2 3 2 4
cluster 1: vertices (1, 4, 6)	*cluster 1*: vertices (1, 2)
cluster 2: vertices (2, 3)	*cluster 2*: vertices (3, 5)
cluster 3: vertices (5, 7)	*cluster 3*: vertices (4)
	cluster 4: vertices (7)

The steps of the heuristic uniform cross-over are given in Table 1.

At each step, one of the uncovered vertices and one of the parents is selected randomly. The uncovered vertices in the cluster which contains the selected vertex in the selected parent are grouped into one cluster and the newly covered

Table 1. Steps of heuristic uniform cross-over

step	chosen vertex	chosen parent	formed clusters	covered vertices
1	4	2	(4)	4
2	1	1	(4)(1,6)	1,4,6
3	7	1	(4)(1,6)(5,7)	1,4,5,6,7
4	3	1	(4)(1,6)(5,7)(2,3)	1,2,3,4,5,6,7

vertices are marked as covered. This continues until all vertices are covered. As can be seen, as a result of this process, the number of clusters in the child may be equal to or different from both of the parents. However, it still contains partial clustering information from its parents.

4.3. The Mutation Operator

A standard mutation operator is used. With a given mutation probability, the value of each gene is reset to a new value within the allowed interval. In the graph clustering problem, this corresponds to removing a vertex from the cluster it is currently in and placing it into another randomly selected cluster. As a result of mutation, some clusters may become empty. In this case, a post-processing step is applied to adjust the new clustering solution.

4.4. Heuristic Disband Operator

The cross-over operator tends to increase the number of clusters. To improve performance the heuristic disband operator is applied to the child. The vertices belonging to the cluster with the minimum intra cluster similarity are placed in the closest cluster. The closest cluster is defined here as the cluster with the maximum average similarity to the vertex.

4.5. Parent Selection

For each iteration of the EA, two parents are selected for reproduction. In this study we used a standard tournament selection method.

4.6. Fitness Evaluation

In this paper, the clustering criteria function is optimized by minimizing the *min-max cut* function [9] and maximizing the silhouette index [19]. Min-max cut function (MMC) combines both the maximization of similarity within each subgraph and minimization of similarity among subgraphs, and is defined as:

$$MMC = \sum_{m=1}^{k} \frac{cut(G_m,\ G \setminus G_m)}{\sum_{v_i,v_j \in Gm} E(v_i, v_j)} \tag{1}$$

where $cut(G_m,\ G \setminus G_m)$ is the sum of edges connecting the vertices in G_m to the rest of the vertices in graph $G \setminus G_m$ and $E(v_i, v_j)$ is the weight of the edge connecting vertices v_i and v_j. The min-max cut function tends to minimize the number of clusters. In order to determine the proper number of clusters adaptively, we use Silhouette index as a second objective. For a given cluster, C_j, $j = 1, ..., k$ the silhouette technique assigns i-th member $(v_i, i = 1, ..., n_j)$ of cluster C_j a quality measure (silhouette index)[19]:

$$s(v_i) = \frac{b_i - a_i}{max\{a_i, b_i\}} \tag{2}$$

where n_j is the number of vertices in cluster C_j, a_i is the average dissimilarity between the vertex v_i and the rest of vertices in cluster C_j, b_i is the minimum dissimilarity between vertex v_i and cluster C_m for $m = 1, ..., k,\ m \neq j$. In this paper the dissimilarity between two vertices v_i and v_j is computed as $1 - E(v_i, v_j)$. The global silhouette value GS is calculated as:

$$S_j = \frac{\sum_{i=1}^{n_j} s(v_i)}{n_j},\ GS = \frac{\sum_{j=1}^{k} S_j}{k} \tag{3}$$

where S_j is the silhouette index of a cluster C_j. The silhouette index takes values between -1 and 1. It is close to 1 when the partition is good and thus the number of partitions (k) is appropriate and close to -1 if not. We combine the two objectives, MMC and GS, by using a weighted sum approach. For simplicity, we take both objectives as maximization and use the fitness function as:

$$fitness = w_1 * 1/(1 + MMC) + w_2 * GS \tag{4}$$

Both objectives balance each other out to find an optimal partition.

5. Experimental Results

There are two aspects to the experimental evaluation of the method proposed in this paper. We begin with a study on synthetic data aimed at evaluation the ability of finding clusters of good quality. The second part of the study focusses on real world data obtained from user accesses on a Web site and assesses whether the proposed technique can be embedded in a sequence space that reveals cluster structure.

For all experiments, the following parameter set, determined empirically, is used for the steady state EA: population size is 50, probability of cross-over is 1, tournament

selection size is 2, chromosome length is equal to the number of vertices, the probability of mutation is *1/chromosome_length*, the probability of heuristic disband is 1 and the weights of the fitness function $w_1 = 1$ and $w_2 = 2$.

For the synthetic data set, the quality of clustering solution is measured by using two metrics that look at class labels of data items assigned to each cluster [22]. The first metric, entropy, measures how the various classes of data items are distributed within each cluster. A low entropy value means, that the data items are clustered effectively. High entropy value, on the other hand, indicates wide divergence in class labels among data items in a cluster. The second metric, purity, measures the extent to which each cluster contains data items from primarily one class [22]. A high purity value means that the data items in one cluster have mostly one of the class labels. In general, larger values of purity means that the clustering solution is better.

5.1. Synthetic Data

We implemented a graph generator to construct a connected, undirected graph with a given number of vertices and partitions. We have apriori knowledge about the class labels of each vertex which enables us to use entropy and purity metrics to evaluate our clustering approach. For our experiments, we generated two graphs with 1000(G_1) and 4000(G_2) vertices for a clustering of 10. The results of those graphs are given in table 2. The results are given for 10 runs of the algorithm and the maximum number of fitness evaluations is set to 10000. The *success rate* shows the percentage of times the global optimum was found. *Avg. evals* and *Std. dev.* are the average and the standard deviation values for the number of fitness evaluations needed to find the global optimum. Finding the global optimum means that the correct number of clusters are identified and the vertices are correctly clustered. Based on the definition for entropy and purity, these values are only given for the runs during which the optimal number of clusters was found. As can be seen from the table, in most runs the global optimum was found through few fitness evaluations. Besides, the algorithm is scalable across graphs with different number of vertices.

Table 2. Results for the clustering solution of the synthetic data sets

	Success Rate	Avg. evals.	Std. dev.	Avg. ent.	Avg. pur.
G_1	0.9	2435	828	0	1
G_2	0.8	2185.1	737.03	0	1

Table 3. Results for the clustering solution of the real data set

	Avg.	Stdev
GS	0.11	0.03
MMC	0.54	0.33
Fitness	1.18	0.047
Evals.	49906	89

5.2. Real World Data

For real world application, we used a data set obtained from Web server access logs of a Web site. The information provided by the Web server can all be used to construct a data model consisting of several user sessions. Since user sessions are ordered URL requests, we can refer to them as sequences of Web pages. The behavior of Web users can be modelled using sequences. The data set is cleaned in order to obtain user sessions where each user session is defined as the sequence of page views for a single visit of a user to a Web site. The cleaning step is beyond the scope of this paper and the details of this step are given in [7]. The pairwise similarities of user sessions are calculated according to the similarity metric proposed in [6, 7]. A graph is constructed whose vertices are user sessions. There is an edge between two vertices if the similarity value between those vertices computed as described in [7] is greater than 0 and this edge is weighted by this similarity value.

The results of this data set is given in Table 3 for 5 runs of the algorithm. The maximum number of fitness evaluations is set to 50000. In 4 of the runs, the algorithm found 5 clusters for the data set and in the remaining run the number of clusters was determined as 4. The Cluto program [5] was run for the data set for 5 clusters in order to compare the clustering solution. The following results are obtained: $GS = 0.02$, $MMC = 0.29$ and $Fitness = 0.33$. The GS for the solutions found by our algorithm is much better than that of Cluto and the MMC values are close. These results show that our algorithm clusters the data set more effectively. Increasing the maximum number of fitness evaluations may improve our results further.

As can be seen from the results, our method is very effective for finding the proper clustering solution. One powerful advantage of our evolutionary graph clustering approach is that our method is able to find the correct number of clusters. Thus, it is very appropriate for the data sets where the structure of the data is unknown and it is very difficult to provide the number of clusters as an input parameter.

6. Conclusion and Future Work

We have considered the problem of clustering sequential data by using a graph model. We introduce a novel graph-based evolutionary sequence clustering algorithm that uses multiple criteria for the evaluation. This leads to robust clustering solutions with respect to the structure of data. This method can effectively determine the number of clusters. We have also presented experimental results both on graphs generated synthetically with different structures and a real world data set.

We are now extending the model in several ways. We will experiment with different representations and genetic operators existing in literature. This may lead to further improvements in our algorithm. We are planning to test the scalability of the algorithm and the effects of the different genetic algorithm parameters in more detail. This clustering algorithm can be applied in many sequence clustering domains; for example we are planning to use it for recommendation models in web usage mining. This is a work in progress and the promising preliminary results promote further study.

Acknowledgements

The authors would like to thank H. Turgut Uyar for his valuable suggestions and ideas during the design phase of the evolutionary representation and operators.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and A. Protasi. *Complexity and Approximation-Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[2] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[3] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, volume 2832, pages 568–579, 2003.

[4] C. K. Cheng and Y. A. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE Trans. on Computed Aided Design*, 10:1502–1511, 1991.

[5] Cluto. http://www-users.cs.umn.edu/ karypis/cluto/.

[6] Ş. Gündüz and E. Adalı. Experimental study of a similarity measure for two dimensional sequences. In *Proc. of the 3rd Asia Pacific Int. Symp. on Information Technology*, pages 383–388, 2004.

[7] Ş. Gündüz and M. T. Özsu. A web page prediction model based on click-stream tree representation of user behavior. In *Proc. of 9. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 535–540, 2003.

[8] Ş. Gündüz Öğüdücü and A. Ş. Uyar. A graph based clustering method using a hybrid evolutionary algorithm. *WSEAS Transactions on Mathematics*, 3(3):731–736, 2004.

[9] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proc. of the 2001 IEEE Int. Conf. on Data Mining*, pages 107–114, 2001.

[10] A. E. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[11] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. E. Arnold, London, UK, 2001.

[12] J. Gasvoda and Q. Ding. A genetic algorithm for clustering on very large datasets. In *Proceedings of International Conference on Computer Applications in Industry and Engineering (CAINE'03)*, pages 163–167, 2003.

[13] J. Handl and J. Knowles. *Multiobjective machine learning*, chapter Multiobjective clustering and cluster validation. Springer, 2005.

[14] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76:175–181, 2000.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[16] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley Sons, Inc., New York, USA, 1990.

[17] S. Kim. *Computational Biology and Genome Informatics*. World Scientific, 2003. Chapter 4.

[18] Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs*. Springer, 1999.

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

[20] N. Selvakkumaran and G. Karypis. Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization. *IEEE Transactions on Computer Aided Design*, 2005. to appear.

[21] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Patterns Analysis and Machine Intelligence (PAMI)*, 2000.

[22] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis, 2001. Tech. Report TR 01–40, Dept. of Computer Science, Univ. of Minnesota, MN, 2001.