A Graph Based Clustering Method using a Hybrid Evolutionary Algorithm

ŞULE GÜNDÜZ ÖĞÜDÜCÜ, A. ŞİMA UYAR
Department of Computer Engineering
Istanbul Technical University
Maslak, Istanbul TR34469
TURKEY
{gunduz,etaner}@cs.itu.edu.tr

Abstract: - Clustering of data items is one of the important applications of graph partitioning using a graph model. The pairwise similarities between all data items form the adjacency matrix of a weighted graph that contains all the necessary information for clustering. In this paper we propose a novel hybrid-evolutionary algorithm based on graph partitioning approach for data clustering. The algorithm is currently tested on synthetic datasets to allow controlled experiments and the results show that our method can effectively cluster data items.

Key-words: Sequence clustering, pairwise similarity, evolutionary optimization, evolutionary algorithms, estimation of distribution algorithms

1 Introduction

The rapid development of computer technology in the last decades has made it possible for data systems to collect huge amounts of data. Analyzing such large datasets is tedious and costly, and thus, we need efficient methods to be able to understand how the data was generated, and what sort of patterns and regularities there exist in the data. A research area in computer science that considers these kinds of questions is called data mining. One of the topics that has been extensively studied in data mining is clustering. Clustering is a discovery process that groups a set of items having similar characteristics. Clustering methods allow users to summarize and organize the huge amounts of data in order to help them in finding what they are looking for.

In this paper, we propose a new graph based clustering method using the criterion function, called min-max cut [3]. The motivation behind this criterion function is that the sequence clustering process can be viewed as partitioning the sequences into subgraphs by minimizing the similarities among subgraphs and maximizing the similarities within each subgraph. The optimal solution to this problem is NP-hard [1], because of the combinatoric nature of the problem. For this reason, evolutionary algorithms (EA) is a good solution approach to optimize this function. To optimize artificial systems, EAs [4, 12] 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. They start with a random set of solution candidates and at each iteration try to improve their quality. Even though normally they are able to deal with large search spaces and are able to produce solutions of acceptable quality in a short time, recently it has become a common practice to hybridize them with other existing approaches to even improve their performance. In this paper, we hybridize a simple EA with an estimation distribution algorithm (EDA) [7] to fasten its first hitting time. Work on EDAs has increased recently and very successful applications can be found in literature. However one of their shortcomings is their tendency to get stuck at local optima. To overcome this, diversity maintaining techniques should be introduced. The hybridization between the EA and the EDA in this paper, aims at combining the powerful features of both approaches. Within our awareness, existing tools for graph partitioning or clustering based on hybrid EAs are hard to find. Therefore, we concentrate in this study on a clustering method based on a hybrid EA that effectively identifies clusters by optimizing the min-max cut function. This approach to graph partitioning is novel and unique. One of the key characteristics of the evolutionary based approach is the criterion function. Our method is easy to adapt to

any criterion function and does not require any single cutoff value. We experimentally evaluated our method using a synthetic data set to be able to really understand how it functions and to be able to perform controlled experiments. We generated our graphs to be highly irregular or random, and to have nodes of dramatically varying degrees. The preliminary experimental results show that using an EA hybrid for graph partitioning improved the quality of the 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. Section 2 describes graph based clustering and optimization methods. Section 3 presents our proposed approach for graph partitioning. Section 4 provides detailed experimental results. In Section 5, we examine related work. Finally, in Section 6 we conclude and discuss future work.

2 Graph Based Clustering

In this study, we focus on pairwise data clustering. Formally, the pairwise data clustering problem is defined as follows: Given $S = (V, N, W, C)$, where V is a set of data items represented as vertices in a graph G, $N \subseteq V \times V$ is a set of data pairs, W is a symmetric matrix of similarity values of each data pair in N, and C is the clustering criteria function, partition the data set V into clusters such that the criteria in C is optimized. Let $P = \{P_1, ..., P_k\}$ be k clusters on the graph G. In this study the clustering problem is formulated as partitioning the graph G into k disjoint subgraphs by minimizing *min-max cut* function [3]. Min-max cut function combines both the maximization of similarity within each subgraph and minimization of similarity among subgraphs, and is defined as:

$$minimize \sum_{m=1}^{k} \frac{c(G_m,\ G \setminus G_m)}{\sum_{v_i, v_j \in Gm} W(v_i, v_j)} \qquad (1)$$

where $c(G_m,\ G \setminus G_m)$is the sum of edges connecting the vertices in G_m to the rest of the vertices in graph $G \setminus G_m$ and $W(v_i, v_j)$ is the weight of the edge connecting vertices v_i and v_j. We often identify a cluster P_i with the induced subgraph $G_i = (V_i, E_i)$, where V_i is the set of vertices in subgraph G_i such that $\bigcup_{i=1}^{k} V_i = V$ and $E_i = \{\{u, v\} \in E \wedge u, v \in V_i\}$. Then the set $E_{in} = \bigcup_{i=1}^{k} E_i$ is the set of *intra cluster edges* and $E_{out} = E \setminus E_{in}$ is the set of *inter cluster edges*.

3 Finding Clusters

The clustering method used in this paper consists of four stages. The general algorithmic flow of the approach can be seen in Figure 1.

```
1:   decrease search space;
2:   randomly initialize EA;
3:   repeat
4:     generate population;
5:     select pairs for reproduction;
6:     recombine pairs;
7:     mutate individuals with pm1;
8:     evaluate new individuals;
9:     estimate probabilities of alleles;
10:  until criteria_met;
11:  restore search space;
12:  initialize EA with seeds from previous stage;
13:  repeat
14:    generate population;
15:    select pairs for reproduction;
16:    recombine pairs;
17:    mutate individuals with pm2;
18:    evaluate new individuals;
19:    estimate probabilities of alleles;
20:  until stopping_criteria_met;
```

Figure 1: Algorithmic flow for hybrid-EA

In the first stage, the search space is reduced by heuristically combining nodes which have higher similarities. The motivation behind this pre-processing step is that nodes that have higher similarities are more likely to end up in the same cluster. By reducing the search space, the running time of the hybrid-EA is reduced as will be seen in the next section. To achieve this reduction, the similarity data are scanned to heuristically combine the nodes. In the second stage, the hybrid-EA is randomly initialized and run on the reduced search space for a predefined number of generations. In the third stage, the combined nodes are separated to restore the search space to its original size. In the final stage, the hybrid-EA is reinitialized using the best solution obtained from the second stage to seed a predefined percentage of the population. These stages will be further explained in the following subsections.

3.1 Search Space Reduction

As expected, an EA based approach performs faster in smaller search spaces. So we pre-processed the similarity matrix and heuristically combined nodes to reduce the running time of the whole algorithm. For ex-

ample if the original graph consists of 4000 nodes, by combining them in groups of four will reduce it to 1000 nodes. Assume that a sub-part of a similarity matrix shows that the similarity values between n1 and five other nodes are as follows:

	n1	n2	n3	n4	n5	n6	...
n1	-	0.6	0.1	0.5	0.7	0.4	...

Based on these similarity values, n1, n2, n4 and n6 will be combined to be labelled as the new n1.

3.2 Search Space Restoration

Before the third stage of the approach, the search space is restored to its original size. The combined nodes are separated and the cluster number of the combined node is assigned to all of them. For example assume that as in the example in the above section n1, n2, n4 and n6 have been combined and labelled as the new n1 and the first application of the hybrid-EA assigned this new n1 to cluster 3. In the restoration stage, the four nodes are separated and each node is assigned to cluster number 3. This heuristic method assumes that similar nodes will be placed in the same cluster, but it aims at maximizing the similarity values within a cluster. However it disregards the second objective of minimizing the cut between clusters. The reason this approach works is that the clustering solution found by the hybrid-EA applied to the reduced search space is only used to seed a percentage of the population in the third stage. Because of this, the hybrid-EA still has the necessary diversity to refine this solution.

3.3 The Hybrid EA Approach

A simple EA approach is hybridized with an EDA [7] like approach. One of the preliminary and simple EDA approaches in literature is the population based incremental learning (PBIL) introduced in [8] and [9]. The original PBIL works by estimating the probability distribution of good alleles in the population and initializing the population at each generation using these distributions. It is originally proposed for binary representations. It incorporates no selection or mutation mechanisms. One of the drawbacks of this approach, especially in multimodal fitness landscapes, is that it has a tendency to get stuck at local optima. A diversity maintaining technique is required to overcome this deficiency. In this paper, we introduce mutation and selection into a PBIL like approach and thus hybridize it with EA like mechanisms.

Each solution candidate in the search space is represented as a haploid individual which means that each individual only has one chromosome. Each gene location on the chromosome represents a node in the graph. The genes can take on integer values showing into which cluster the corresponding node is placed. For example, for a graph with 5 nodes, the following individual depicts a solution that places the first and the third nodes in the first cluster, the second node in the second cluster and the fourth and the fifth nodes in the third cluster.

individual:	1 2 1 3 3

The fitness of the individual shows how good a solution it is. The objective function given in Equation 1 is for minimization. However for the PBIL part of the hybrid approach we need to transform it into maximization. Assume that the original objective function in Equation 1 is named as *obj*, the fitness function we used for the maximization is given in the below equation as follows:

$$fitness = 1/(1 + obj) \tag{2}$$

There is a global probability matrix that shows the probability of each gene g_i to be initialized to each allele a_j at the beginning of each generation. For example assume for a chromosome length of five and three alleles, the current probability matrix is as follows:

	g_1	g_2	g_3	g_4	g_5
a_1	0.4	0.0	06	0.3	0.8
a_2	0.2	0.0	0.1	0.3	0.0
a_3	0.4	1.0	0.3	0.4	0.2

Using this matrix, at the beginning of the current generation g_3 for example will be initialized to a_1 with probability 0.6, to a_2 with probability 0.1 and to a_3 with probability 0.3.

The global probability matrix is calculated at the end of each generation using a percentage of the best offspring individuals. For example for allele a_j and gene location i, the probability value $prob_{ji}$ will be calculated as given in equation 3:

$$prob_{ji} = \frac{\sum_{m=1}^{n} f_m}{\sum_{k=1}^{n} f_k} \; for\ all\ chromosome_{mi} = a_j \tag{3}$$

where n is the total number of chromosomes in the population that are used in the calculation of the probability values, f is the fitness value of each chromosome, $i = 1, ..., chromosome\_length$,

$j = 1, ..., allele\_count$. Based on this, the $prob_{ji}$ value will be higher if individuals with the allele a_j in the ith location have higher fitnesses compared to those that have other alleles at that location.

As can be seen in Figure 1, after the population is initialized at the start of each generation, the rest of the loop runs like a regular EA. For the selection of parents, a tournament selection with tournament sizes equal to $1/5$ of the population size is used. Uniform cross-over with a predefined probability is applied to selected parents and a uniform mutation is applied to each gene with probability equal to $1/chromosome\_length$. The offspring replace the parents and the new probability matrix values are calculated using this offspring population. The main loop of the algorithm is run for a predetermined number of generations.

The same hybrid-EA approach is used for both stage 2 and stage 4. The only difference is that for stage 2, the population is fully randomly initialized. However for stage 3, 10% of the population is initialized to the clustering solution produced as a result of stage 2.

4 Experimental Results

We conduct our experiments on synthetic data aimed at evaluation the ability of finding clusters of good quality. For evaluating the quality of a clustering solution, two different well known metrics are used, namely entropy and purity [11]. The quality of clustering solution is measured by using two metrics that look at class labels of data items assigned to each cluster. The first metric, entropy, measures how the various classes of data items distributed within each cluster. Given a particular cluster P_m, the entropy of this cluster is defined to be [11]:

$$E(P_m) = -\frac{1}{log\ k} \sum_{i=1}^{k} \frac{n_m^i}{n_m}\ log \frac{n_m^i}{n_m} \tag{4}$$

where k is the number of classes in the data set, n_m is the number of data items assigned to the mth cluster, and n_m^i is the number of data items of the ith class that were assigned to the mth cluster. The entropy of the entire clustering solution is then defined as the sum of the individual entropies of each cluster weighted by the number of data items assigned to the cluster:

$$Entropy = \sum_{m=1}^{k} \frac{n_m}{n} E(P_m) \tag{5}$$

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 extend to which each cluster contains data items from primarily one class. The purity of a cluster P_m is defined as the fraction of the number of data items of the cluster to the largest number of data items assigned to that cluster [11]:

$$Pr(P_m) = \frac{1}{n_m} max_i(n_m^i) \tag{6}$$

A high purity value means that the data items in one cluster have mostly one of the class labels. The overall purity of the clustering solution defined to be:

$$Purity = \sum_{m=1}^{k} \frac{n_m}{n} Pr(P_m) \tag{7}$$

In general, larger values of purity means that the clustering solution is better.

We implement a graph generator to construct a graph with a given number of partitions. Given the number of the vertexes n and the number of partitions k, each vertex is assigned to one of the partitions. Next, a uniformly random clustered graph is generated by inserting intra cluster edges with probability p_{in} and inter cluster edges with probability p_{out}. In case a graph constructed that way is not connected, additional edges are added to connect the components. We can assume that for a graph generated that way the initial clustering has the expected behavior with respect to the values of p_{in} and p_{out}. Thus, we have a prior knowledge about the class labels of each vertex which provides to use entropy and purity metrics to evaluate our clustering approach.

For our experiments, we set the number of vertexes $n = 4000$, the number of partitions $k = 10$, $p_{in} = 0.7$ and $p_{out} = 0.3$. For this graph ($G_0 = (V_0, E_0)$), we run the experiments in two levels. Instead of trying to compute partitioning directly in the original graph, we

[1]Our clustering algorithm is programmed in C language without any code optimization. Further experiments on computation time will be conducted in future.

first obtain a sequence of graphs whose size is smaller than the size of the original graph. The size of the original graph is decreased by merging vertices as mentioned in Section 3. The motivation behind this process is to shorten runtime for the clustering programs[1]. In case of clustering a smaller graph of 1000 nodes ($G_1 = (V_1, E_1)$) one fitness calculation is 2000 times faster then the program which clusters a graph of 4000 nodes. Due to lack of space, we just present the results of the experiments which are performed in two levels.

For the experiments performed in the tests, the following parameter settings for the hybrid-EA were used: population size is 50, probability of cross-over is 0.8, tournament selection size is 10, number of individuals used in global probability matrix calculation is 10, chromosome length is equal to the number of nodes, the probability of mutation is *1/chromosome_length* and the maximum number of generations is set to 3000 and 7500 for the 1000 nodes and 4000 nodes instances respectively.

The results of those experiments are presented in Table 1. The experiments are performed 20 times and the entropy and purity values are calculated for each clustering solution. Furthermore, the number of generations are considered for each clustering solution. The results in Table 1 show the average value of fitness, entropy, purity and number of generations as Avg. Fit., Avg. Ent., Avg. Pur. and Avg. Gen. respectively. In addition to those results the standard deviation of fitness value and number of generations are calculated. The average of entropy and purity values are only calculated for the final clustering solutions. The fitness value of the generated graph should be 0.338. As can be seen from the Table, the average value of fitness is 0.22, which confirms that our clustering solutions are quite effective. Furthermore, the results show that the clustering solutions have very low entropy values and high purity values as desired.

Table 1: Results for the clustering solution obtained via two level partitioning

	Avg. Fit.	Std. Fit.	Avg. Gen.	Std. Gen.	Avg. Ent.	Avg. Pur.
G_1	-	-	4593.6	1281.1	-	-
G_0	0.220	0.096	4698.6	2102.6	0.065	0.957

Over the entire set of experiments, the smallest entropy value is 0.022. In this case the fitness value is 0.306. Figure 2 shows the confusion matrix for the experiment that has the best solution for entropy and purity. The confusion matrix can be used for determining the misclassification. For example the first cluster has 452 items and 447 of those are from the same class (i.e. only 5 items are misclassified). As can be seen from the Figure, all of the clusters contain items that mostly belong to the same class.

0	0	0	1	1	0	2	447	1	0
0	1	0	0	0	1	0	3	0	410
1	398	0	0	0	0	0	0	0	0
0	0	0	0	0	388	0	3	0	0
1	0	0	0	0	0	388	0	1	0
0	0	0	0	1	1	0	0	388	1
0	0	375	0	0	0	0	0	0	0
408	0	1	0	0	2	1	0	1	0
0	0	0	362	0	0	0	2	0	0
0	0	1	1	403	0	3	0	1	1

Figure 2: Confusion Matrix

The results confirm that our clustering method can effectively cluster data items. Besides this, reducing the search space decreases the running time of the clustering algorithm.

5 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 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 [5, 10] 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 comparing to the other subgraphs. Various constraints are introduced, such as ratio cut [2], normalized cut [6], and min-max cut [3], etc. to remedy this problem.

6 Conclusion and Future Work

We have considered the problem of clustering data items by using a graph model. We introduce a novel hybrid-evolutionary algorithm based on graph partitioning approach for data clustering. Results show that hybrid-EA gives good clustering. Our proposed method provides good preliminary results to promote our further study. Instead of trying to compute partitioning directly in the original graph, we reduce the search space by merging the nodes in the graph. Thus, the running time of the clustering algorithm is reduced dramatically. However, the program runtime is still long, which would be shorten in the future by programming optimization.

We are now extending the clustering method in several ways. Experimentally, we have demonstrated the success of hybrid EA by clustering data items. We are planning now to experiment the method with other hybrids. The proposed method, however, is general and can be applied in a variety of graph partitioning problems. We will try our method on real world data obtained from user accesses on a Web site in order to embed it in a Web page recommendation model. All this will be the subject of future work.

References

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

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

[3] Zha H. Gu M Simon H. Ding C, He X. 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.

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

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

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

[7] Larranaga P. and Lozano J. A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computing*. Kluwer, 2002.

[8] Baluja S. Population based incremental learning: A method for integrating genetic serch based function optimization and competitive learning, 1994. Tech. report no: CMU-CS-94-163, Carnegie Mellon Univ.

[9] Baluja S and Caruana R. Removing the genetics from the standard genetic algorithm. In *Proc. of Int. Conf. on Machine Learning*, pages 38–46. Morgan Kaufmann, 1995.

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

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

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