A Distributed Genetic Algorithm for Graph-based Clustering

Krisztian Buza, Antal Buza, and Piroska B. Kis

Abstract Clustering is one of the most prominent data analysis techniques to structure large datasets and produce a human-understandable overview. In this paper, we focus on the case when the data has many categorical attributes, and thus can not be represented in a faithful way in the Euclidean space. We follow the graph-based paradigm and propose a graph-based genetic algorithm for clustering, the flexibility of which can mainly be attributed to the possibility of using various kernels. As our approach can naturally be parallelized, while implementing and testing it, we distribute the computations over several CPUs. In contrast to the complexity of the problem, that is NP-hard, our experiments show that in case of well clusterable data, our algorithm scales well. We also perform experiments on real medical data.

Key words: Graph-based Clustering, Genetic Algorithms

1 Introduction

Since the middle of the twentieth century, computers are applied for data analysis and decision support. In some cases, like systems that select which products should be advertised for users, an automatic decision, without any detailed explanation, might be sufficient. In more serious tasks, such as the

Krisztian Buza
Information Systems and Machine Learning Lab, University of Hildesheim,
Marienburger Platz 22, D-31141 Hildesheim, Germany, e-mail: buza@ismll.de and
Dpt. of Information Theory and Computer Science, Budapest Univ. of Technology and Economics, H-1117 Budapest, Magyar tudósk körútja 2., Hungary, e-mail: buza@cs.bme.hu

Antal Buza and Piroska B. Kis
College of Dunaujvaros,
Tancsics Mihály u. 1/a, H-2400 Dunaujvaros, Hungary, e-mail: {buza,piros}@mail.duf.hu

ones in engineering, industry or medicine, a human-understandable explanation is crucial in order to justify semi-automatic decisions, and in order to turn the data into stable, transferable and well-founded knowledge. One of the most prominent analytic tasks, that contribute to knowledge discovery in the above sense, is clustering. In case of *clustering*, the computer is aimed at structuring a large set of data by producing groups of similar objects. From the user's perspective, such groups, called *clusters*, allow for more understandable and more transparent representation of a large datasets.

Throughout the analytic process, man-machine interaction is crucial in at least two steps: (i) when the user describes her requirements to the computer (e.g. what kind of groups would be beneficial in the underlying application, when should two objects be considered to be similar), and (ii) when the result of the analysis is presented to the user. In many cases, the user is unable to formulate her requirements in a completely exact way (e.g. in terms of logical formulas), but she has some intuition, e.g. regarding the number of objects in a group and variance of their attribute-values, etc. Through the usage of kernels, our approach allows the user to set such requirements for (a) the clustering as a whole, (b) each cluster, and (c) the similarity of two objects.

In the most simple (and most studied) case, a dataset consists of vectors of *real numbers*. Such data is usually considered as points in the Euclidean space. In contrast, in many applications (e.g. medical and psychological surveys), large amount of *categorical* attributes are present, and therefore the data can not be represented in a natural and faithful way in the Euclidean space.

In this paper, we focus on the above case and follow the graph-based data representation paradigm [9]. In particular, we propose a flexible graph-based genetic algorithm for clustering. Kernels, that allow for flexibility, are one of the core components of our approach. Therefore, we study kernels and identify the class of effective kernels (limited change kernels). As our approach can naturally be parallelized, while implementing and testing it, we distribute the computations over several CPUs. In contrast to the complexity of the problem, which is NP-hard, experiments show that in case of well clusterable data, our algorithm scales well. We also perform experiments on real medical data that provides further evidence about the applicability of our approach.

2 Related Work

By developing our algorithm and especially by its thorough analysis, we aim at better understanding clustering problems in general. Such goal has been followed by many authors recently. After Kleinberg [10] introduced his criteria for clustering, Ackerman and Ben-David [2] presented a refined theory. The stability of clustering was analyzed in e.g. [3] and [4], while Shamir and Tishby related the rate of convergence to model selection [13]. From the complexity point of view, several clustering problems were shown to be

NP-hard. In order to alleviate computational expenses, sampling was proposed [8], [11], [12], [14]. Ackerman and Ben-David developed the notion of 'clusterability' and showed, that "the more clusterable a data set is, the easier it is (computationally) to find a close-to-optimal clustering of that data" [1].

As an alternative of the widely-used vector representation, algorithms working on graph-based representation were proposed [9]. Most closely related to our work is Brown's genetic algorithm for clustering [6] which, similarly to our approach, uses an external objective function. However, in contrast to [6], we do not focus on the chemical domain, furthermore our distributed implementation is a unique property of our approach.

3 Graph-based Genetic Algorithm for Clustering

High dimensional Euclidean spaces suffer from many problems, the conglomeration of them is known as the curse of dimensionality. These problems are amplified in case of categorical, due to the presence of a natural and faithful mapping to ordered values, see [9] for an excellent illustration.

Therefore, we follow the graph-based data representation paradigm [9], where each record (object) of the original data corresponds a vertex of a graph and *similar* objects are connected by edges. (We define *similarity* later.) In our approach, we build on the *genetic strategy*, as genetic algorithms are powerful in finding (approximate) solutions to optimization and search problems [5], which is also justified by the fact in AusDM 2009, a recent data mining challenge, a genetic algorithm based solution won, see also: http://www.tiberius.biz/ausdm09/AusDM09EnsemblingChallenge.pdf).

Basically, our approach searches for an appropriate set of cutting vertices nodes in the graph representing the data. This search is performed with a genetic algorithm. After removing the nodes of the found cutting set, the components of the remaining graph correspond to the clusters.

What an ideal clustering is, highly depends on the underlying application. In order to allow for generality, in our algorithm, the ideal clusters can be characterized with an *external cluster quality function*, that we call *kernel*. This is in line with the direction followed in [6].

Definitions, Notations and Problem Formulation. Let c be a function which assigns a positive real number to a sub-graph (a potential cluster). Let h be a function which assigns a real number to any set of real numbers, e.g., $h(\{1,3,10\}) = 8$. Functions c and h together constitute our two-component *clustering kernel*, i.e. the external quality function used to measure the goodness of a clustering: first the function c is applied for each cluster, and the returned values are aggregated by h. Then quality of the clustering is characterized by the value returned by h. (We suppose that the functions c and h assign greater value to better clusters and clusterings, respectively.)

Each set of vertices corresponds to a clustering of the graph: the vertices of the set are removed, the remaining components constitute the clusters. The clustering task is to find the cutting set of vertices at which the function h has its maximal value: Let g_0 denote the graph that is to be clustered. Let $V(g_0)$ be the set of vertices of g_0. Let $g_1 C g_2$ denote that the graph g_1 is a component of the graph g_2. Let $X \subset V(g_0)$. Let $g_0 \backslash X$ denote the graph which is derived from graph g_0 by removing the vertices in X. (As usual, the corresponding edges are also removed.) The *clustering problem* is to search for the set of nodes $X \subset V(g_0)$ that maximizes h:

$$\text{argmax}_X(h(\{c(g')|g' C (g_0 \backslash X)\})). \tag{1}$$

We introduce the concept of *limited change kernels (LCK)* which we use in our analysis later on. If a cutting vertex set gives a good clustering, a slightly different vertex set is also likely to give a good clustering. Therefore, the cluster quality, i.e. the value returned by h should be similar for similar vertex sets, which allows for the genetic algorithm to find a close-to-optimum solution efficiently. Let us call the operation of inserting or removing an element into or from a set as an *editing step*. The distance between two sets m_1 and m_2, denoted by $dist(m_1, m_2)$, is the minimal number of the editing steps required to transform m_1 to m_2. Given a graph g_0 a clustering kernel consisting of the functions c and h is a *limited change kernel*, if for any number $\epsilon > 0$ there is a *finite threshold* d so that:

$$\forall m_1 \subset V(g_0), \forall m_2 \subset V(g_0) : \tag{2}$$

$$dist(m_1, m_2) < \epsilon \implies |h(\{c(g')|g' K (g_0 \backslash m_1)\}) - h(\{c(g')|g' K (g_0 \backslash m_2)\})| < d$$

Note, that Equation 2 generalizes the notion of continuous functions for the case of graph kernels. Throughout the paper, we denote the number of vertices and edges in a (sub)graph g with $|V(g)|$ and $|E(g)|$ respectively.

Complexity analysis.

Theorem 1. *The clustering problem (see Equation 1) is NP-hard.*

Proof. In a complete graph, two arbitrary vertices are connected by an edge. Let function c be the following: $c(g) = \begin{cases} |V(g)| & \text{if the graph } g \text{ is complete} \\ 0 & \text{else} \end{cases}$

Let h be the maximum-function: $h(\{x_1, x_2, ..., x_n\}) = max(x_1, x_2, ..., x_n)$.

We allow the cutting vertex set to be the empty set. This way the "clique-problem" (search for the maximal full sub-graph) is reduced to our clustering problem: h is maximized exactly when one of the clusters is the maximal clique. If an algorithm solving our problem outputs a clustering where one of the clusters corresponds to the maximal clique, one can easily select this cluster in a postprocessing step. As the "clique-problem" is NP-complete [7], our problem is NP-hard. (Note that even if the cutting vertex set is not

allowed to be the empty set, the clique-problem can still be reduced to our problem by simply checking in the first step whether g is complete.) □

In general, special cases of NP-hard problems may be much simpler. Our problem, however, is NP-hard even in case of limited change kernels:

Theorem 2. *Let* $c(g) = \begin{cases} 1 - \frac{1}{|V(g)|} & \textit{if the graph } g \textit{ is complete} \\ 0 & \textit{else} \end{cases}$
and let h *be the maximum-function. This is a limited change kernel.*

Proof. For any graph $g, 0 \leq c(g) < 1$ and $0 \leq h(...c(g)...) < 1$. Thus, any $d > 1$ is an appropriate change threshold in Equation 2. □

Corollary: On the analogy of Theorem 1, using the kernel of Theorem 2, the clique-problem can be reduced to our clustering problem (Equation 1). Thus our clustering problem is NP-hard for limited change kernels too.

Our approach: Genetic algorithm for clustering. From now on, we assume that the clustering kernel is a limited change kernel.

Genetic algorithms iteratively simulate the biological evolution. We followed one of the usual ways to start this process: we begin with a population of randomly generated individuals. Individuals are vertex sets in our case. The fitness of a given individual is the value returned by the clustering kernel when partitioning the graph according to that individual. In each iteration the fitness is calculated for each individual of the population. In order to form the new population (i) according to their fitness some individuals are selected from the current population, and (ii) descendants of the selected individuals are formed (by their recombination and mutation). In the first step we select the N best individuals from the current population of $2N$ individuals. In the second step we form further N descendants of the selected individuals. Then the new population (for the next iteration) consists in total of $2N$ individuals again. The algorithm terminates, if the best individual becomes stable, i.e. the same individual is the best for k_{stop} iterations.

The *descendant* s_3 of two individuals (vertex sets) s_1 and s_2 is computed as follows. In the description, g_0 denotes the graph to be clustered:

1. Put all vertices of $s_1 \cap s_2$ into s_3.
2. Each vertex in $s_1 \backslash s_2$ and $s_2 \backslash s_1$ have the same chance to be included in s_3, i.e. put each vertex from $s_1 \backslash s_2$ or $s_2 \backslash s_1$ into s_3 with a probability of 0.5. This way the algorithm is unbiased with respect to the size of sets.
3. *Add/remove some random vertices to/from* s_3. First, we decide whether add or remove vertices, the probability of both cases is 0.5. Then each $v \in V(g) \setminus s_3$ will be added (removed respectively) with probability p.

Steps 1 and 2 realize crossbreeding, step 3 realizes mutation.

According to our observation, the most time-consuming phase in the genetic algorithm is the calculation of fitness, i.e. the calculation of the value of

the clustering kernel for each individual. In our implementation, we parallelized these computations by letting different CPUs to calculate it for different individuals in a distributed environment.

Note that we are concerned with the exploration of the high-level structure of the data (graph) by finding characteristic groups of objects (vertices). For this reason, we allow our to omit the vertices in the cutting vertex sets. If, however, it is relevant to which clusters (components) these objects (vertices) belong, they can be assigned to clusters in a postprocessing step.

4 Experiments

Scalability experiments. In the first experiment, our approach was tested on three types of artificial benchmark graphs: (i) The *star* consists of k complete graph components of the same size and an additional vertex, called central vertex, that is connected with one vertex from each component. (ii) The *tricky star* is similar to star, but all vertices of all components are connected to the central vertex. (iii) The *ring* consists of k complete graph components of the same size and additional k edges that connect these components.

For each of the benchmarks we performed experiments in two versions: in the first case the number of components k was varied at fixed component-size of 20, while in the second case we varied the size of the components at fixed $k = 5$. We set the parameters of the genetic algorithm $N = 200$, $p = 0.2$, $k_{stop} = 10$ and used the following clustering kernel:

$$c(g) = \frac{2|E(g)|}{|V(g)|(|V(g)|-1)} + \left(1 - \frac{1}{|V(g)|}\right) \quad , \quad h(\{x_1, ..., x_n\}) = \frac{x_1 + ... + x_n}{n}$$

In all these test cases the algorithm found an appropriate cutting vertex set. Table 1 shows the numbers of iterations of the genetic algorithm averaged over 3 runs. Although the set of all possible solutions grows exponentially, we observed moderate growth in the number of required iterations.

Table 1 Average number of generations in the genetic algorithm

Size of the graph[a]	Graph Type					
	STAR I	T.STAR I	RING I	STAR II	T.STAR II	RING II
100	21.00	19.33	18.66	20.00	19	28.5
200	25.00	24.67	32.66	26.00	24	33
300	27.67	27.00	41.00	27.00	25	40
400	29.67	28.33	43.66	30.00	30	40.5
500	31.00	29.33	56.00	30.50	30.5	40.5
...	...	...	...	...	...	...
1000	36.00	36.00	44.33	36.50	35	53

[a] total number of vertices, without the central vertex in case of Star and TrickyStar

Experiments on real data. For our experiments we used a subset of WHO Europe's dataset of the Countrywide Integrated Noncommunicable Diseases Intervention (CINDI). This contains persons' answers to various questions about their health status. We selected those 887 persons who did not visit the doctor and the attributes that were most interesting for the domain expert like the indicators for high blood pressure or diabetes.

In this case, vertices of the graph correspond to persons. In order to determine the similarity of two persons (which allows us to decide whether or not they are connected in the graph), we used the following formula:

$$s(r_1, r_2) = \frac{\sum_{i \in C} f(i, r_1, r_2) + \sum_{j \in I} (1 - \frac{|r_1.j - r_2.j|}{|max(j) - min(j)|})}{|C \cup I|}$$

where C is the set of categorical attributes; I is the set of numeric attributes; r_1, r_2 are records of the database (persons); $max(j), min(j)$ stands for the maximal/minimal value of the numerical attribute j and $f(i, r_1, r_2) = 1 - \frac{v}{n_{size}}$ if the values of the i-th attributes of the records r_1 and r_2 are equal; $f(i, r_1, r_2) = 0$ otherwise; here v is the number of the records for which the value of the attribute i is equal to $r_1.i$ and n_{size} is the total number of the records in the data set. Thus, for categorical attributes: a rare value being equal for two objects counts more, as if a frequent value would be equal.

In the graph we connected two vertices (persons) if $s(r_1, r_2) > 0.1$, which resulted in total in almost 200,000 edges. We used the following kernel:

$$c(g) = \frac{2|E(g)|}{|V(g)|(|V(g)| - 1)} + \frac{|V(g)|}{4000} \quad , \quad h(\{x_1, ..., x_n\}) = \frac{x_1 + ... + x_n}{n}$$

Further parameters of our algorithm were set to $N = 500, p = 0.2, k_{stop} = 10$.

Our algorithm produced an easily interpretable clustering. Three clusters were identified, the fist one corresponds to healthy persons, the second one corresponds to persons having at least one disease (in the most cases backache), in the third cluster we found persons that consequently did not answer to any questions. We clustered the same data with k-Means and FarthestFirst from WEKA (http://www.cs.waikato.ac.nz/ml/weka/) as well. Using k-Means one of the clusters was poorly established. FarthestFirst categorized almost all the patients into one cluster, except 17 patients. Thus our clustering algorithm promisingly outperformed these two methods.

5 Conclusions

In this paper we proposed a new clustering method based on the search for cutting vertex sets using a genetic algorithm, that can easily be customized for various tasks by the choice of clustering kernel. We analyzed the complexity

of our problem, experimentally investigated the scalability of our method and demonstrated its applicability on real-world data. Graph-based clustering is not limited to the studied problem, therefore we hope that it will be applied in other domains in the future.

References

1. Ackerman, M., Ben-David, S.: Which data sets are clusterable?-a theoretical study of clusterability (2008). URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.3549&rep=rep1&type=pdf`
2. Ben-David, S., Ackerman, M.: Measures of clustering quality: A working set of axioms for clustering. In: Advances in Neural Information Processing Systems 21, pp. 121–128 (2009)
3. Ben-David, S., Pál, D., Simon, H.: Stability of k-means clustering. In: 20th Annual Conference on Learning theory, pp. 20–34. Springer (2007)
4. Ben-David, S., Von Luxburg, U.: Relating clustering stability to properties of cluster boundaries. In: International Conference on Computational Learning Theory (COLT) (2008)
5. Beyer, H.: The theory of evolution strategies. Springer (2001)
6. Brown, N., McKay, B., Gilardoni, F., Gasteiger, J.: A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. Chemical Information and Computer Sciences **44**(3), 1079–1087 (2004)
7. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms. The MIT Press / McGraw Hill (2003)
8. Czumaj, A., Sohler, C.: Sublinear-time approximation algorithms for clustering via random sampling. Random Structures & Algorithms **30**(1-2), 226–256 (2007)
9. Guha, S., Rastogi, R., Shim, K.: Rock: A robust clustering algorithm for categorical attributes. Information Systems **25**(5), 345–366 (2000)
10. Kleinberg, J.: An impossibility theorem for clustering. In: Advances in Neural Information Processing Systems 15, p. 463 (2003)
11. Meyerson, A., O'Callaghan, L., Plotkin, S.: A k-median algorithm with running time independent of data size. Machine Learning **56**(1), 61–87 (2004)
12. Mishra, N., Oblinger, D., Pitt, L.: Sublinear time approximate clustering. In: 12th Annual ACM-SIAM Symposium on Discrete algorithms, pp. 439–447. Society for Industrial and Applied Mathematics (2001)
13. Shamir, O., Tishby, N.: On the reliability of clustering stability in the large sample regime. In: Advances in Neural Information Processing Systems 21, pp. 1465–1472 (2009)
14. de la Vega, W., Karpinski, M., Kenyon, C., Rabani, Y.: Approximation schemes for clustering problems. In: 35th Annual ACM Symposium on Theory of Computing, pp. 50–58. ACM (2003)