AutoClustering

An Estimation of Distribution Algorithm for the Automatic Generation of Clustering Algorithms

Aruanda S. G. Meiguins, Roberto C. Limão
PPGEE - UFPA
Belém, Brasil

Bianchi S. Meiguins, Samuel F. S. Junior
ICEN- UFPA
Belém, Brasil

Alex A. Freitas
University of Kent at Canterbury
Canterbury, UK

Abstract— **Most of the existing Data Mining algorithms have been manually produced, that is, have been developed by a human programmer. A prominent Artificial Intelligence research area is automatic programming – the generation of a computer program by another computer program. Clustering is an important data mining task with many useful real-world applications. Particularly, the class of clustering algorithms based on the idea of data density to identify clusters has many advantages, such as the ability to identify arbitrary-shape clusters. We propose the use of Estimation of Distribution Algorithms for the artificial generation of density-based clustering algorithms. In order to guarantee the generation of valid algorithms, a directed acyclic graph (DAG) was defined where each node represents a procedure (building block) and each edge represents a possible execution sequence between two nodes. The Building Blocks DAG specifies the alphabet of the EDA, that is, any possibly generated algorithm. Preliminary experimental results compare the clustering algorithms artificially generated by AutoClustering to DBSCAN, a well-known manually-designed algorithm.**

Automatic Programming, Density-Based Clustering, Estimation of Distribution Algorithms, Data Mining

I. Introduction

Clustering is an important Data Mining task that groups items in a dataset into meaningful classes. Many clustering algorithms have been proposed [1], [2], [3], [4], [5], [6], [7], [8]. Density-based clustering algorithms, in particular, focus on the identification of clusters of arbitrary shape by identifying dense areas in a dataset. Each algorithm, however, is based on a different definition of density and uses a different process to identify these dense areas.

A common characteristic of all clustering algorithms and actually of most algorithms in general, is that they are manually produced, i.e. they were developed by one or more human beings. A current research topic in Artificial Intelligence is the study of automatic programming. This task is usually the focus of Genetic Programming algorithms. The more recent area of Estimation of Distribution Algorithms (EDA) relies on the evolution of probabilistic models of the analyzed population of individuals instead of the more common evolutionary operators such as mutation and crossover [11].

EDAs have recently been proposed as a new paradigm for automatic program construction [15]. However, in that work the EDA was used to evolve solutions to relatively simple Genetic Programming benchmark problems, i.e., no algorithm invention was attempted, nor any Data Mining task investigated. By contrast, this work addresses the much more challenging task of developing an EDA for the automatic creation of clustering algorithms.

More precisely, the goal of this paper is to develop an EDA called AutoClustering that automatically generates density-based clustering algorithms. It is important to distinguish the goal of this work from other projects using evolutionary algorithms for clustering. The output of those other evolutionary algorithms was a set of clusters selected for a specific dataset [10].

The present work has a much more ambitious goal. We propose the evolution of a new clustering algorithm, specifically designed for a given dataset. Therefore, the output of AutoClustering is a fully-fledged clustering algorithm – including loops, for instance.

To the best of our knowledge there is no previous work on the automatic generation of clustering algorithms. The most related work seems to be [16], which proposes a genetic programming system for automatically generating a very different kind of data mining algorithm, viz. classification algorithms. Classification involves supervised learning, unlike clustering, which involves unsupervised learning, so the two tasks are very different. Also, we propose an EDA for our target task, whilst [16] proposed a very different genetic programming system.

In order to increase the chances for a good clustering algorithm to be generated by AutoClustering, the alphabet is defined as a number of carefully-selected typical clustering procedures (building blocks of clustering algorithms). Additionally, the EDA uses a pre-defined data structure that implicitly represents all the valid sequences of building blocks.

This data structure is a directed acyclic graph (DAG), which specifies the generic structure of density-based clustering algorithms. We will call this a Building Blocks DAG. AutoClustering searches the Building Blocks DAG after the execution of each building block to recall which blocks are valid next.

We have analyzed in detail many density-based clustering algorithms, in order to identify the most significant density-based clustering procedures. Many recently proposed algorithms present relatively small contributions in relation to previous algorithms. We have therefore selected a small number of basic density-based clustering algorithms – those presenting a particularly different approach to the task. Any subsequent version of these algorithms proposed later for slightly improving performance or flexibility was therefore not considered.

As expected, the selected basic algorithms had some building blocks in common. A set of typical density-based clustering procedures was then extracted from these basic algorithms. The procedures were generalized as building blocks to be executed in any order specified by a given sequence of edges in the Building Blocks DAG. Hence, the building blocks will be combined by the EDA producing new density-based clustering algorithms. As a result, these automatically-created clustering algorithms will be potentially free from the limitations and biases associated to the manually-created existing clustering algorithms, introducing a new level of automation in the clustering task of Data Mining.

The next sections will present the selected basic clustering algorithms, the corresponding identified building blocks, the Building Blocks DAG and the proposed structure for the EDA.

II. Selected Algorithms

Eight density-based clustering algorithms were selected as basic algorithms for this work.

- DBSCAN [2] is a well-known density-based clustering algorithm. It identifies clusters as a set of density-connected items in a dense area. It uses the parameter MinPts to specify the minimum number of items for a certain neighborhood to be considered dense and the parameter Eps to specify the radius of this neighborhood. Items not included in any cluster are labeled as noise.
- DENCLUE [3] initially partitions the dataset into multidimensional hyper-cubes. It identifies the local density function associated to each item in the highly populated cubes and its adjacent cubes. A hill-climbing procedure determines the local maximum for each item – called density-attractor. A cluster is then defined for each density-attractor item and its attracted items.
- DBCLASD [7] increments an initial cluster while the internal distance is compatible to the expected distance distribution. SAMs (spacial access methods) are used to fetch the surrounding items.
- CLIQUE [1] searches for dense units in a bottom-up approach starting with one-dimensional dense units. It generates a partition of connected units in the dataset.
- DHC [4] produces an attraction tree connecting each item to its density attractor. The attraction tree is reduced to a density tree that identifies the clusters.
- DESCRY [6] builds an adaptable k-d tree from the dataset. Each node of the tree is associated to approximately the same number of items. Clusters are initially defined as the gravity center of each node and then an agglomerative hierarchical procedure is performed to obtain the number of clusters specified by the user.
- SUDEPHIC [8] partitions the dataset in equally sized grids. The cells with highest density are hierarchically merged. The criterion for the cells to be merged is the density in the overlap area.
- AMR [5] creates well-spaced grids from the dataset and recursively refines the highly dense regions. This process generates a hierarchical tree representing the dataset. Each node of the tree is initially considered a cluster. The algorithm then analyzes the parent nodes of each cluster assigning each item to the closest cluster.

III. Proposed Building Blocks

The analysis of the basic algorithms briefly referred to in the previous section inspired the design and implementation of the following set of 16 density-based clustering building blocks.

Some of these building blocks are used in more than one of the selected clustering algorithms, but only one algorithm is specifically indicated in the description of the building block for the sake of simplicity. The implementation of the building block was based on that indicated algorithm.

Each building block demands the specification of one or more parameters. The parameter values were included in the individual specification so that the EDA would select not only a set of building blocks but also the appropriate parameters in each case.

- **createCandidatesByDistance** – creates cluster candidates formed by items in a specified neighborhood as in the DBSCAN algorithm. Input: a dataset. Output: a set of items. Parameters: the distance, the minimum number of items.
- **createClusterByConnectiveness** – checks if every item in the cluster is density-connected as defined in the DBSCAN algorithm. Input: a set of items Output: a cluster. Parameters: the distance, the minimum number of items.
- **createCandidatesByNPts** – selects the N closest items to each item of the dataset as in the DBSCLAD algorithm. Input: a dataset. Output: a set of items. Parameter: the number of items.

- **createClusterByDistribution** – iteratively adds items to a cluster as in the DBCLASD algorithm. Each item is added to the cluster if the updated cluster distribution is still acceptable. If a not acceptable distribution is achieved, the item is not added to cluster. Input: a set of items. Output: a cluster. Parameter: the number of items.

- **createAttractionTree**– creates a tree connecting each item in the dataset to its attractor as defined in the DHC algorithm. Input: a set of items. Output: an attraction tree. Parameters: the number of points, the sigma value, the threshold value.

- **createDensityTree**– transforms an attraction tree into a density tree as in the DHC algorithm. Input: an attraction tree. Output: a density tree. Parameters: the number of points, the sigma value, the threshold value.

- **createASH**– builds average shifted histograms to identify dense areas in the dataset as in the DENCLUE algorithm. Input: a dataset. Output: a set of items in the dense areas. Parameters: the sigma value, the epsilon value.

- **createClusterByAttractor** – creates a cluster for each density-attractor item found by a hill-climbing procedure as in DENCLUE. Input: a set of items. Output: a set of clusters. Parameters: the sigma value, the epsilon value.

- **idDenseAreas** – recursively identify dense areas in each dimension starting with one-dimensional dense units as in CLIQUE. Input: a set of items. Output: a set of dense units. Parameters: the number of slices, the threshold value.

- **createClusterByPartition** – connects dense units into partitions describing a cluster for each identified partition as in CLIQUE. Input: a set of dense units. Output: a set of clusters. Parameters: the number of slices, the threshold value.

- **createEquallySizedGrid** – divides the dataset into a grid of uniformly sized cells as in AMR. Input: a dataset. Output: a set of cells. Parameter: the number of slices.

- **createAdaptableKDTree**– partitions the dataset into a binary tree where each node is connected to approximately the same number of items as in DESCRY. Input: a set of items. Output: an adaptable kd-tree. Parameters: the number of items in each node, the K value.

- **createAMRTree** – creates a tree of recursively refined grids as in the AMR algorithm. Input: a set of cells forming a grid. Output: AMR tree. Parameters: the density value, the lambda values, the number of slices.

- **createAMRCluster** – creates a cluster for each leaf of the input AMR tree and assigns parent items to the corresponding cluster in a bottom-up agglomerative process. Input: an AMR tree. Output: a set of clusters. Parameters: the density value, the lambda values, the number of slices.

- **mergeByOverlap** – hierarchically merges similar clusters until the specified number of clusters is reached, using the density in the clusters overlap as similarity criterion as in the SUDEPHIC algorithm. Input: a set of partitions for a dataset. Output: a set of clusters. Parameters: none.

- **mergeByDistance** – merges the closest pair of clusters until the specified number of clusters is reached as in the DESCRY algorithm. Input: a set of partitions for a dataset. Output: a set of clusters. Parameter: the number of clusters.

IV. INDIVIDUAL REPRESENTATION

An individual for the proposed EDA is a density-based clustering algorithm. The algorithm is any valid sequence of the selected building blocks, that is, a possible sequence from the Building Blocks DAG in Fig. 1, from the root node to a terminal node in the figure, plus a set of parameter values for all the parameters of the building blocks composing the sequence.

Hence, the Building Blocks DAG incorporates important background knowledge about the clustering task that supports AutoClustering in the task of creating valid clustering algorithms.

Each node of the Building Blocks DAG represents one of the density-based clustering building blocks. Each directed edge represents a sequential connection between two procedures, that is, the procedure pointed by the edge should be executed immediately after the execution of the procedure from which the edge comes out. Each edge has an associated weight that represents the probability of the procedure pointed to by the edge to be selected. The edge weights are not shown in the figure in order to keep it simple.

The total weight (sum of probabilities) of the edges coming out from a node is always 1, since those edges represent mutually exclusive ways of selecting a new procedure to be executed next, but the individual probabilities of the edges are updated each time a new population is produced by the EDA.

This probability update follows a conventional scheme for updating the probability vector of an EDA like PBIL [11], with the difference that in our case each component of the "probability vector" is the probability associated with an edge (connecting two procedures) in the Building Blocks DAG, rather than being a variable as in PBIL.

More formally, an EDA individual is represented by a path in the Building Blocks DAG from the initial node to a terminal node, i.e., a node from which no edge comes out. A path in the Building Blocks DAG is a sequence of nodes n1,..., nk where n1 is the initial node, ni, i = 2,...,k is a child (successor) node of ni - 1 and nk is a terminal node.

Recall that an individual represents a density-based clustering algorithm. Different clustering algorithms are generated for each different path in Building Blocks DAG.

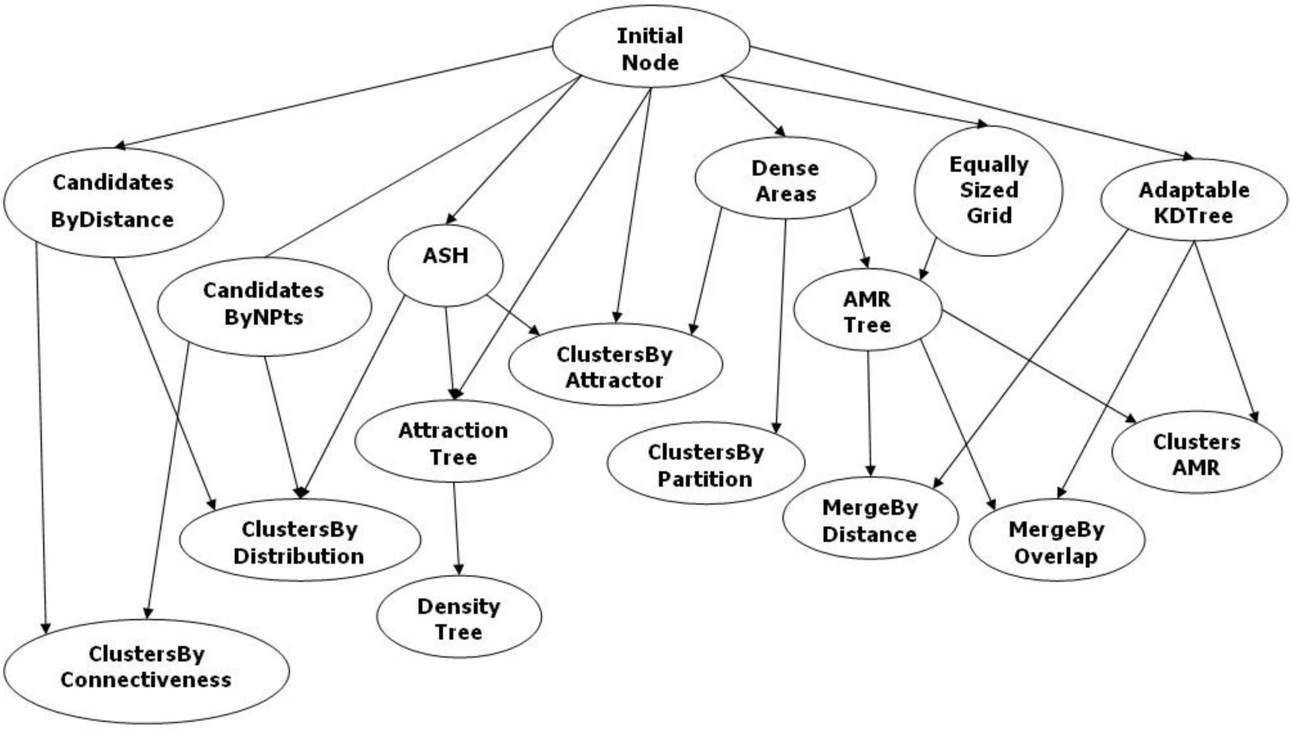

Figure 1. Building Blocks DAG

For instance, a candidate algorithm for a particular dataset could be represented by the sequence of nodes in Fig. 2. This path would represent a novel clustering algorithm inspired by a combination of ideas from three existing density-based clustering algorithms: CLIQUE, AMR and DESCRY.

Additionally, the evolved algorithm would include the best-suited parameter values for the specific dataset involved. The parameter values are part of the individual representation and are evolved by the EDA. For each building block associated to each individual, the necessary parameter values are selected from a pre-specified range of values. For instance, the number of slices for the AMR algorithm may vary from 25 to 30.

There are 6,051,800 different possible algorithms that may be generated for each dataset considering the 20 possible paths of the building blocks DAG and the possible parameter values for each building block.

V. Fitness Function

The fitness of an individual must be based on the results of the corresponding clustering algorithm. However, defining the fitness function is not a trivial task, since clustering is an unsupervised learning task. The user does not supply previously clustered data to the clustering algorithm and therefore, although there are several objective criteria to evaluate a clustering result [10], the ultimate evaluation of a clustering process is usually regarded as a subject issue. This is in contrast with classification algorithms, which have well-defined objective evaluation criteria, since they are supervised learning algorithms. Note that in this work it is essential to use an objective criterion to evaluate clustering results, since we are automating the process of clustering algorithm creation. Our proposed solution to this problem is described next.

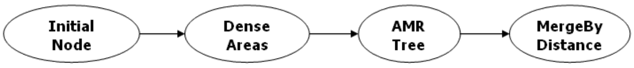

Figure 2. Possible Path in Building Blocks DAG

In general, the result of a clustering algorithm can be expressed as a list of data items where each item is associated to a specific cluster or, in some cases, to no cluster (when the item is considered noise). In order to evaluate the results of the clustering algorithms produced by AutoClustering we propose the use of a method that takes advantage of the objective nature of classifier evaluation.

The Clest method [9] was originally proposed to estimate the adequate number of clusters for a dataset. We have adapted this method to estimate the quality of clustering results in the following steps.

1. Split the dataset in two subsets: a training set and a test set.
2. Apply the clustering algorithm to the training set.
3. Assign a label to each data item indicating the cluster it belongs to.
4. Apply a classification algorithm to the training set using as target (class) attribute value the cluster label specified by the clustering algorithm. This will generate a classifier, which was trained from this specific subset of items.

5. Apply the clustering algorithm to the test set specifying a cluster (target attribute value) to each item.
6. Apply the classifier built in step 4 to the test set.
7. Compare the results of the clustering algorithm and the results of the classifier on the test set.

The evaluation of an AutoClustering individual therefore considers the performance of the classifier on the test set, after the clustering algorithm has been independently used to assign cluster labels to data items in both training and test sets.

The main advantage of the proposed adapted Clest method over existing objective metrics for clustering evaluation [13] lies in the flexibility and generality of the former approach. Any classifier may be used to evaluate the clustering results. Therefore, this method is applicable to any dataset with any combination of data types. Other proposed clustering evaluation metrics would introduce a strong bias towards some distribution of the data or particular data types.

The fitness of an individual takes into consideration the performance of the corresponding clustering algorithm on the given dataset. The fitness function value for each individual is determined by the accuracy rate of the classifier on the test set, after the corresponding clustering algorithm has specified cluster labels to the training and the test sets.

VI. AUTOCLUSTERING – AN EDA FOR AUTOMATIC DENSITY-BASED CLUSTERING ALGORITHM GENERATION

The EDA uses a probability DAG instead of the more commonly used probability vector. The evolution process updates the probabilities associated to each edge of the BuildingBlocksDAG according to the evaluation of the algorithms using the corresponding building blocks.

Fig. 3 presents the pseudo-code for updating the probability DAG. In this procedure, α is the average fitness of selected individuals using the corresponding edge.

The population evolution is therefore guided by the probabilities of the BuildingBlocksDAG edges. The better the individuals using a specific edge, i.e., using both the building blocks coming from and to the edge, the better are the chances of this edge to be selected by the EDA.

```
Parameter DagNode – A node in a probability
directed acyclic graph
Parameter SelectedIndividuals – a subset of
individuals from the current population selected
by any evolutionary selection method (default: 50%
highest-fitness individuals).
Procedure updateDagProbabilities (DagNode,
SelectedIndividuals)
For each Edge coming out from current node
  edgeProb= α + (1 – α)EdgeProb;
  updateDagProbabilities (target node of current
                      edge, selectedIndividuals);
totalProb = total EdgeProb for all Edges coming
            out of current node;
For each Edge coming out from current node
    EdgeProb = EdgeProb / TotProb;
```

Figure 3. Updating the edge probabilities in the BuildingBlocksDAG

The following steps outline the execution of AutoClustering.

1. Generate a population of N density-based algorithms. Each individual (or algorithm) represents a possible path in the BuildingBlocksDAG.
2. Evaluate each individual using the CLEST method.
3. Eliminate the 50% individuals with lowest fitness.
4. Update the probabilities of each edge of the BuildingBlocksDAG according to its frequency of use in the selected population (the 50% highest-fitness individuals).
5. Generate a new population of N individuals using the updated BuildingBlocksDAG
6. Add the best individual of the previous population to the current one (implementing elitism).
7. Repeat steps 2-7.

VII. EXPERIMENTAL RESULTS

Current experiments have been based on four public-domain datasets from the well-known UCI dataset repository, often used for benchmarking in machine learning research [12]: Glass Identification (8 attributes, 214 instances); Pima Indians Diabetes (8 attributes, 768 instances); Bupa Liver-disorders (7 attributes, 345 instances); and Cleveland Heart Diseases (13 attributes, 294 instances). The class attributes of each dataset were not considered.

The EDA was set to evolve 50 individuals during 500 generations. In order to compare results of the best algorithm produced by the EDA to the manually-designed selected algorithms, we have set Autoclustering to exclusively run pre-selected paths of the Building Blocks DAG. So it was possible to reproduce the exact sequence of procedures of the original algorithms and therefore obtain the results of CLIQUE, AMR, DESCRY, DBSCAN, DBCLASD, DENCLUE, SUDEPHIC and DHC for the selected datasets.

We then performed 30 executions of the EDA and 30 executions of each of the 8 manually-designed algorithms for each dataset in order to analyze the difference in accuracy between the results of the algorithms generated by AutoClustering and the other evaluated algorithms.

The same Clest method used in the fitness of the EDA was used when running the evaluated algorithms, in order to make the comparison between these algorithms as fair as possible.

Each of the evaluated algorithms independently produced cluster labels in the training and test subsets for each dataset, as explained earlier. The J48 classification algorithm [14] (a Java version of the well-known C4.5 decision-tree induction algorithm) built a classifier based on each training set. The classifier was applied to the corresponding test set for each of the selected datasets. We compute the accuracy between the cluster label produced by the clustering algorithm and the corresponding classifier label, for each data item in the test set.

We then compare the algorithms in terms of the accuracy rate of the comparison between the two labels for all data items in the test set. The box plots in the following figures indicate the minimum and maximum accuracy rate values, the corresponding lower and upper quartiles and the medium for each of the four selected datasets.

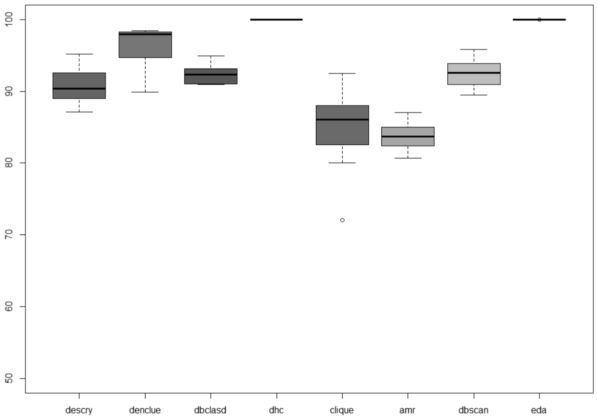

Figure 5. Boxplot – fitness values on executions using the Bupa dataset

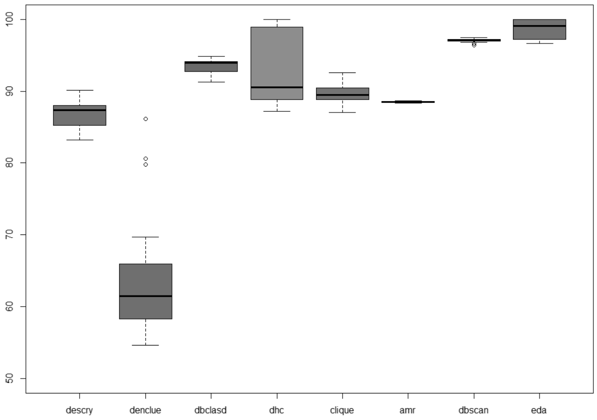

Figure 6. Boxplot – fitness values on executions using the Cleveland dataset

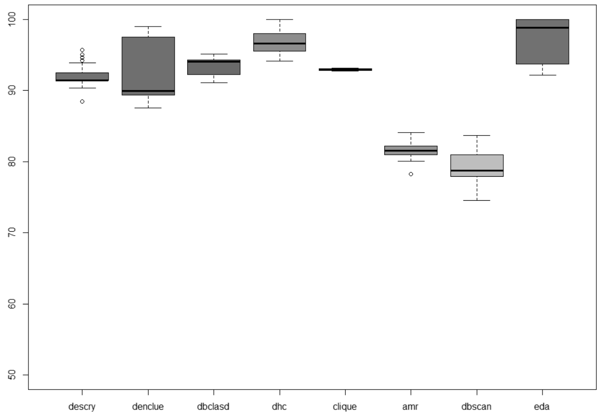

Figure 6. Boxplot – fitness values on executions using the Pima dataset

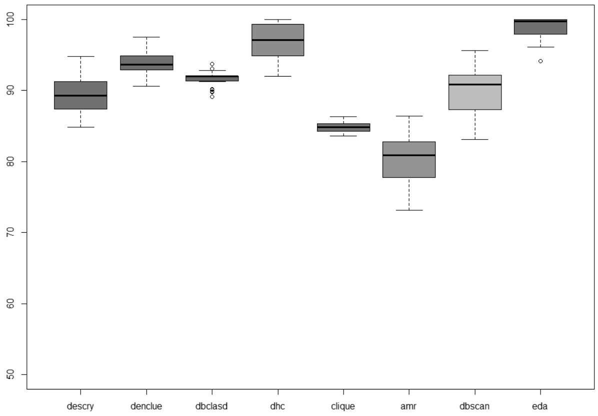

Figure 4. Boxplot – fitness values on executions using the Glass dataset

The automatically-evolved algorithms by our EDA presented the best results for each dataset. DHC, DBCLASD and DBSCAN also presented good results for the Cleveland dataset. DENCLUE, DBCLASD and DHC performed well when analyzing the glass dataset. Only AMR and DBSCAN produced results below 80% when analyzing the Pima dataset. Finally, when analyzing the Bupa dataset, the best basic algorithms were DESCRY, DENCLUE, DBCLASD and DBSCAN.

Autoclustering has built 30 different algorithms for each of the four analyzed datasets. In some cases, the only variation was in terms of parameter values since the same path was selected on the building-blocks DAG. In other cases, the selected path represented one of the classic clustering algorithms, such as DENCLUE or DBSCAN. However, in most cases the selected path on the building-blocks DAG represented a combination of parts of different algorithms, that is, essentially new algorithms – never before evaluated in any previous work.

One simple example of an algorithm generated automatically by autoclustering is represented in Fig.8. This algorithm was evolved for the PIMA dataset. It begins by identifying dense areas as in the CLIQUE algorithm and finishes with the DENCLUE procedure that identifies clusters by the attraction criterion.

VIII. CONCLUSIONS

In this paper we propose the use of an EDA for automatic generation of density-based clustering algorithms. Many existing clustering algorithms were reviewed so that the appropriate building blocks – typical clustering procedures – could be identified.

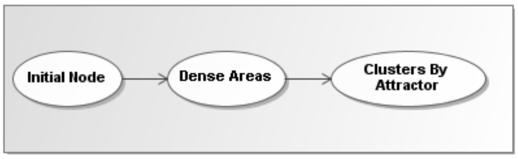

Figure 8. Path of one of the algorithms generated by Autoclustering for the Pima dataset.

The definition of the Building Blocks DAG specified the possible sequence of building blocks covered by the EDA, that is, the possible artificially generated algorithms.

The fitness of an individual is based on an adaptation of the Clest method. The generated clustering algorithm is executed on both training and test sets separately, so the appropriate cluster label is assigned to each data item. Then a classifier algorithm is executed on the training set and applied to the test set. The fitness is then determined by the accuracy of the classifier on the test set. We emphasize that information about the classes of examples in the test set is not used during the training of the classification algorithm, preserving the fundamental principle of separation between training and test sets in the classification task. The idea is that a more suited clustering algorithm for a given dataset will produce cluster labels that are more easily predicted by a classifier method because they represent a consistent structure in the dataset.

The algorithms generated by Autoclustering for the selected public-domain datasets performed better, on average, than any of the 8 selected classic clustering algorithms. The developed EDA will potentially generate efficient algorithms specifically designed for any particular dataset.

This approach is particularly interesting when one needs to repeatedly apply a clustering algorithm to the same dataset, so the computation cost is compensated by the benefits of generating a new clustering algorithm tailored to the dataset.

Our plans for future work include the analysis of the difference in accuracy between the results of the classic algorithms and the algorithms generated by AutoClustering. We intend to use ANOVA, a statistical test that generalizes the Student t-test for more than two groups.

REFERENCES

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos and Prabhakar Raghavan, “Automatic subspace clustering of high dimensional data” in ACM-SIGMOD Int. Conf. Management of Data. Washington, 1998.

[2] M. Ester, H-P. Kriegel, J. Sander and X. Xu “A density-based algorithm for discovering clusters in large spatial databases with noise”, in Proc. of KDD-96, AAAI Press, 1996. pp. 226 - 231.

[3] A. Hinneburg and D. Keim, “A general approach to clustering in large databases with noise”, in Knowledge and information systems, vol. 5, n. 4., Springer London, 2003, pp. 387 - 415.

[4] D. Jiang, J. Pei and A. Zhang, “DHC: A Density-based Hierarchical Clustering Method for Time Series Gene Expression Data”, in Proceedings of the 3rd IEEE Symposium on Bio-informatics and Bio-engineering, USA, pp. 393-400, 2003.

[5] W. Liao, Y. Liu and A. Choudhary, “A Grid-based Clustering Algorithm using Adaptive Mesh Refinement” in Proceedings of the 7th Workshop on Mining Scientific and Engineering Datasets, Lake Buena Vista, USA, April 2004.

[6] F. Angiulli, C. Pizzuti and M. Ruffolo, “DESCRY: A Density Based Clustering Algorithm for Very Large Dataset”, Fifth International Conference on Intelligent Data Engineering and Automated Learning, Exeter, UK, 2004.

[7] X. XU, M. Ester, H-P Kriegel, J. Sander. “A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases” in Proceedings of the Fourteenth International Conference on Data Engineering.Orlando, USA, 1998. p. 324 - 331.

[8] D. ZHOU, Z. Cheng, C. Wang, H. Zhou, W. Wang and B. Shi, “SUDEPHIC: Self-tuning Density-based Partitioning and Hierarchical Clustering” in Lecture Notes in Computer Science, 2004, vol. 2973, pp. 69-108.

[9] S. Dudoit and J. Fridlyand, “A prediction-based resampling method for estimating the number of clusters in a dataset” in Genome Biology, vol. 3, n. 7, pp. 1-21, 2002.

[10] A. A. Freitas,. Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer, 2002.

[11] J. A. Lozano, P. Larrañaga, I. Inza and E. Bengoetxea. Towards a new Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer-Verlag, 2006. 294p.

[12] A. Frank and A. Asuncion, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2010.

[13] P-N. Tan, M. Steinbach, V. Kumar Introduction to Data Mining. Boston: Addison Wesley, 2005. 769 p.

[14] I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kauffmann Publishers, San Francisco, 2000.

[15] K. Yanai and H. Iba, “Estimation of Distribution Programming: EDA-based approach to program generation” in J.A. Lozano et al. (Eds.) Towards a new evolutionary computation: advances on EDAs, 103-157. Springer, 2006.

[16] G.L. Pappa and A.A. Freitas. Automating the Design of Data Mining Algorithms: an evolutionary computation approach. 187 pages. Springer, 2010.