# PASCAL: An EDA for Parameterless Shape-Independent Clustering

Henry E. L. Cagnini
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Rio Grande do Sul, Brazil
Email: henry.cagnini@acad.pucrs.br

Rodrigo C. Barros
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Rio Grande do Sul, Brazil
Email: rodrigo.barros@pucrs.br

***Abstract*—Data clustering is an unsupervised learning task that can be regarded as the combinatorial optimisation NP-hard problem of assigning $N$ objects to one (or more) among $k$ clusters. Most data clustering algorithms require the user to set a number of pre-defined parameters that have decisive impact in the formation of clusters, such as the number of clusters (initial or final), cluster radius, minimum number of objects, and similar parameters. In addition, several clustering algorithms are limited with regard to the shape of clusters that can be found, a limitation usually resulting from the optimisation process performed over a given distance metric. In this work, we propose a novel clustering algorithm that addresses the two aforementioned problems regarding the amount of parameters and cluster shape. Our approach makes use of the theory of Estimation of Distribution Algorithms in order to probabilistic sample a set of must-link/cannot link constraints in order to generate a data partition with the proper number of clusters. We name our method PASCAL, and we empirically show that it is capable of not only detecting the right number of clusters but also of properly assigning objects to the correct cluster in a variety of artificial and real problems whose solutions are known in advance.**

***Keywords*—estimation of distribution algorithms, clustering, machine learning**

## I. Introduction

Clustering is an important task within machine learning, consisting of finding groups of similar objects within a batch or stream of data. From an optimisation viewpoint, it consists of assigning objects to a given number of clusters ($k$) in order to maximise the intra-cluster similarity and minimise the inter-cluster similarity. Note, however, that there are $\frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^N$ possible assignments of $N$ objects to $k$ clusters [1]. The notion of quality regarding clustering solutions is somewhat subjective and often domain specific, even though there are different criteria that can be used to estimate the overall quality of a partition under some prior assumption.

Each clustering algorithm adopts a particular strategy as well as a set of assumptions for generating partitions or hierarchies of partitions. Partitional clustering methods such as the well-known $k$-means algorithm [2] usually optimise a given cohesion criterion (e.g., the Sum of Squared Errors) via an iterative optimisation procedure. Hierarchical agglomerative clustering algorithms [3], [4], in turn, generate a hierarchy of partitions ranging from the trivial solution of one cluster per object to the total merge of the entire dataset into a single cluster. A typical strategy for analysing these algorithms is to build a dendrogram representing the step-by-step process of merging clusters, and by making horizontal cuts in this dendrogram one can extract up to $N$ partitions (considering $N$ the number of objects in the dataset) with varying number of objects per cluster.

Even though several well-known clustering algorithms implement a particular greedy heuristic for choosing a near-optimal solution, it is only natural that data clustering be approached by global-search meta-heuristics having in mind the fact that it is a NP-hard task [3]. Hence, several evolutionary algorithms for clustering have been proposed throughout the years, and we refer the interested reader to the thorough survey by Hruschka et al. on the matter [5]. More recently, with the rise of the Estimation of Distribution Algorithms (EDAs) [6], which achieved state-of-the-art performance in a variety of distinct optimisation problems, we believe its application in the domain of clustering offers exciting reseearch possibilities.

EDAs work based on three main aspects: encoding of individual, probabilistic model (along with a sampling strategy), and fitness function. An EDA holds a set of possible solutions (individuals) which comprise a population. It starts by sampling individuals from an uniform distribution encoded within a probabilistic model, which maps the relationship among variables. Once sampled, each individual has its fitness asserted through a domain-specific criterion that assesses the quality of the solution. From a limited set of best individuals the distribution is updated in every generation and thus novel individuals are sampled, with the process continuing until a maximum number of generations (or convergence) is reached.

In this paper, we present an estimation of distribution algorithm to perform clustering in arbitrarily-shaped datasets, namely PASCAL (PArameterless Shape-independent Clustering ALgorithm). The individual encoding in PASCAL presents two advantages: (1) it prevents the generation of a *round search space* by making sure that different genotypes are mapped to different phenotypes, which is a problem that related work fail to prevent; and (2) since it is based on a minimum spanning tree to encode its probabilistic graphical model, it starts from a promising region in the solution space that is much more likely to present better results than otherwise. PASCAL addresses two major issues in data clustering, which is the ability to handle clusters with different shapes (e.g., other than spherical), and the fact that it does not require the setting of clustering-related parameters (e.g., number of clusters, radius, minimum density, etc.). We empirically analyse

978-1-5090-0623-6/16/$31.00 

PASCAL's performance in 9 artificial datasets whose cluster labels are known in advance, and in one real-world labeled dataset in which we assume the mapping class-to-cluster holds. We show in our experiments that PASCAL is capable of not only correctly predicting the number of groups in the data, but also of properly assigning objects to their corresponding clusters.

The remaining of this work is organised as follows. Section II introduces the reader to our novel method for data clustering. Section III reports the methodology that was followed in the empirical analysis for validating our approach, allowing for reproducibility. Section IV describes the results of the experiments and a discussion on our findings. Section V presents a brief literature review on related work. Finally, we draw our conclusions and point to future work in Section VI.

## II. PASCAL

Consider a set of objects $o_1, o_2, \cdots, o_N \in O$ in a $\Re^m$ dimensional feature space. For assigning $N$ objects to $k$ clusters, there are a total of $\frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^N$ possibilities [1]. Hence, the challenge in data clustering is not only to correctly estimate the number of clusters $k$, but also to properly assign each object to these clusters.

One strategy to reduce the combinatorial problem without loss of quality is by taking into account geometrical aspects of the data. Let us assume that all objects are connected to each other by $E$ edges. An edge weight is given by the distance between a pair of objects (say the Euclidean distance, without loss of generality). Hence, we have a dense graph $G$ with objects representing the vertices. If we now consider that each edge links two objects that belong to the same cluster, the task of finding clusters is reduced to the one of removing unnecessary edges from the graph. This graph modelling maps the set of assumptions in PASCAL: defining each edge of the graph as either a must-link or a cannot link relationship. The next section goes into further details of each step of PASCAL's evolutionary clustering.

### A. Encoding Individuals as MSTs

It is fairly intuitive that one does not need to start from a fully-connected graph in order to remove the edges that will most probably maximise a given clustering quality criterion. Instead, by realising that far-away objects are much less likely to belong to the same cluster than closer objects, one could start with a graph whose edges only connect an object to its closest neighbour. Indeed, the strongest link of an object $o_i$ is with its nearest neighbour, and if not even that neighbour is in the same cluster than $o_i$, then no other object will be (this is only true of course under the assumption that minimising the distance is the equivalent of maximising similarity).

Hence, instead of building a fully dense graph, PASCAL builds a minimum spanning tree (MST), which connects the set of $N$ vertices with a subset $E_\pi$ of $N-1$ edges from $E$ that minimise the overall weight of the tree while preventing cycles. The distance measure used for calculating the weight of edges can be any one which satisfies the symmetry ($D(x_i, x_j) = D(x_j, x_i)$), positivity ($D(x_i, x_j) \geq 0\ \forall i, j \in [0, \ldots, N]$), and reflexivity ($D(x_i, x_j) = 0$ iff $x_i = x_j$) properties [3]. A well-known distance measure that satisfies all these properties plus the triangle inequality ($D(x_i, x_j) \leq D(x_i, x_k) + D(x_k, x_j)$ for all $x_i, x_j$ and $x_k$) is the Euclidean distance, which is used as the default distance measure in PASCAL.

PASCAL employs the Kruskal's algorithm [7] for generating the MST based on a pre-computed distance matrix. Kruskal's algorithm starts by considering each vertex as a candidate subtree, and also all possible edges in the graph. It then iteratively removes an edge with minimum weight from the set of edges, and if that edge connects different trees it combines them into a single tree. At the end of the process, since the graph is connected the single resulting tree is an MST containing all $N$ vertices (i.e., objects) and $N-1$ edges.

PASCAL starts the search of clusters within the MST by considering which edges should be disconnected and which should not. Naturally, there are $2^{(N-1)} - 2$ valid clustering partitions to be found in this search, which means the number of solutions grows exponentially with the number of objects. Using an MST presents an advantage over the label-based approaches usually employed in evolutionary clustering algorithms (e.g., [8]): it prevents the search space to be artificially enlarged due to multiple genotypes that are actually mapped to the same phenotype. Figure 1 exemplifies this situation: three candidate solutions that can be found by a label-based evolutionary algorithm have the same cluster assignment (phenotype), but three different labels for objects (genotypes).

| | | | |
|---|---|---|---|
| $P_1$ | 1 | 2 | 3 |
| $P_2$ | 3 | 1 | 2 |
| $P_3$ | 2 | 3 | 1 |
| | $o_1$ | $o_2$ | $o_3$ |

Fig. 1: Three individuals ($P_1, P_2$ and $P_3$) with different label assignments for objects $o_1, o_2, o_3$, but with the same genotype (cluster assignment).

Once the MST is built, it is mapped into a probabilistic graphical model (GM) [9]. A GM is a resource used by EDAs to sample new individuals for comprising the population of candidate solutions. The relationship among variables with regard to their mutual interaction dictates the type of GM: it may be either direct, in which a children variable does not affect the outcome value of the parent variable; or indirect, in which this interaction is possible. Even though MSTs are intrinsically undirected, we make use of direct inference for sampling values from probabilities, and hence we convert the MST into a direct GM. Such a transformation is straightforward, with objects becoming variables and edges becoming the probability of two objects being in the same cluster. The initial probability of objects $o_i$ and $o_j$ to belong to the same cluster $C_l$ is given by Equation 1:

$$p(o_i, o_j \in C_l) = 1 - \frac{d(o_i, o_j)}{\sum_{e \in E_\pi} w_e} \quad (1)$$

where $d(o_i, o_j)$ is the distance between objects $o_i$ and $o_j$ and $w_e$ is the weight of edge $e$ that belongs to the set of edges

$E_\pi$ of the MST. Hence, closer objects have a higher initial probability to be in the same cluster than farther ones.

Once we have the MST encoding every pair of nearest objects with the corresponding probability of each pair to be in the same cluster, we can sample from a univariate marginal distribution in order to identify which pairs of objects in an individual will hold a **must-link** constraint and which will hold a **cannot link** constraint. Hence, each individual in PASCAL is a list of boolean constraints indicating whether the objects linked in the MST should or should not be in the same cluster. Note that each pairwise constraint is considered independent of one another, so the probability of a pair of objects being in the same cluster will not affect the constraints of another pair, giving PASCAL enhanced agility and speed when sampling individuals and updating the distribution.

Note that PASCAL is completely parameterless regarding the clustering process: one does not need to specify a value of $k$ clusters nor any initial probability of clustering a pair of objects together, although that could be easily arranged by modifying components such as the distance matrix, the MST edges, or the GM probabilities. The only parameters used by PASCAL are those that control the search in an EDA: population size, maximum number of iterations (total budget allowed), fraction of fittest individuals that update the GM probabilities, and fraction of individuals to be replaced in the following generation. We did not perform any attempt to tune these search parameters during the experimental analysis that will be presented in Section IV. In fact, we argue that PASCAL is robust and insensitive to these parameter choices in terms of cluster quality, and that well-established common-sense values for these search parameters are rather sufficient for performing high-quality clustering with PASCAL. Its pseudocode is presented in Figure 2.

```
1: function PASCAL(data, iter, frac_induce, frac_replace)
2:     compute the distance matrix from data
3:     build MST from data
4:     initialise GM with objects as variables, edges from MST as relationships, and
       edge weights as probabilities
5:     sample an entire population with GM
6:     evaluate population
7:     while iter > 0 and GM has not converged do
8:         update GM based on frac_induce individuals with best fitness
9:         remove frac_replace of worst individuals from the population
10:        sample frac_replace individuals from GM and add to the population
11:        evaluate population
12:        iter ← iter − 1
13:    return best individual from the population
```

Fig. 2: Pseudo-code of PASCAL.

### B. Fitness function

PASCAL makes use of the density-based clustering validation criterion, DBCV [10], as its fitness function. By using a density-based criterion, PASCAL is capable of detecting arbitrarily-shaped clusters, since its assumption regarding what a cluster is will be based on the concept of finding dense areas separated by sparse regions. As pointed out by Moulavi et al. [10], previously developed density-based criterion fail in several aspects such as correctly analysing arbitrarily-shaped clusters due to the use of the Euclidean distance, which favours the generation of spherical groups, or requiring a parameter such as the number of nearest neighbours to calculate the density of a region where an object lies. DBCV, on the other hand, is a parameterless criterion which defines a new distance to calculate such density.

The calculation of the DBCV index starts by taking as input the cluster assignment for each object. It then computes $a_{pts}coredist$, which is the inverse of density of the object in its cluster:

$$a_{pts}coredist(o_i) = \left( \frac{\sum_{j=2}^{n_i} \left( \frac{1}{d(o_i,o_j)} \right)^m}{n_i - 1} \right)^{-\frac{1}{m}} \tag{2}$$

where $m$ is the dimensionality of the data, $n_i$ the number of objects in the $i$-th cluster and $d(o_i, o_j)$ the distance between objects $o_i$ and $o_j$. Once all objects have their $a_{pts}coredist$ computed, it builds a mutual reachability matrix of the $C_i$ cluster objects using the definition of mutual reachability distance:

$$d_{mreach}(o_i, o_j) = \max(a_{pts}coredist(o_i), a_{pts}coredist(o_j), d(o_i, o_j)) \tag{3}$$

From the $mreach$ matrix of each cluster, a $MST_{MRD}$ is built, which captures the underlying structure of the data. Using each $MST_{MRD}$ it is possible to calculate the Density Sparseness of a Cluster, $DSC(C_i)$, which is the edge with maximum weight in $MST_{MRD}$, and Density Separation of a Pair of Clusters, $DSPC(C_i, C_j)$, which is the minimum mutual reachability distance that separates the inner nodes (e.g nodes with degree 2 or above) of two $MST_{MRD}$.

The validity of a cluster is calculated using both $DSC$ and $DSPC$ values:

$$VC(C_i) = \frac{\min_{1 \le j \le l, j \neq i}(DSPC(C_i, C_j)) - DSC(C_i)}{\max\left(\min_{1 \le j \le l, j \neq i}(DSPC(C_i, C_j)), DSC(C_i)\right)} \tag{4}$$

Finally, the index can be calculated as the weighted average of each $VC(C_i)$:

$$DBCV(C) = \sum_{i=1}^{k} \frac{|C_i|}{|O|} VC(C_i) \tag{5}$$

where $k$ is the number of clusters, $|O|$ is the total number of data objects, and $VC(C_i)$ the validity of cluster $C_i$. The index ranges from $-1$ to $+1$, going from partitions with sparse and overlapping clusters to partitions with dense and well-separated groups, respectively.

### C. Time complexity

PASCAL's time complexity is computed as follows. The calculation of the distance matrix between $N$ objects takes $O(N^2)$. Finding the MST through Kruskal's algorithm takes $O(N \log N)$. The main loop of the EDA runs in the worst case for $T$ times, where $T$ is the number of max iterations. The number of individuals to have its fitness calculated is proportional to the fraction used as input for the EDA, $I$. Since

the whole population must be sampled in the first iteration, it has a complexity of $O(I(1+T))$. The worst case for calculating DBCV is when all objects belong to the same cluster, since $a_{pts}coredist$ will be calculated using all $N-1$ objects as neighbours and at most $N-2$ objects will be an inner node for finding the MST. Hence, it has a complexity of $O((N-2)^2+(N-2)\log(N-2)+N+1)$, which corresponds respectively for calculating $a_{pts}coredist$, finding the MST of the trivial cluster, finding $DSC$, and calculating the DBCV index itself. Updating the GM based on fittest function takes $O(F)$, where $F$ is the number of fittest individuals. Hence, our algorithm has a complexity of $O(N(1+\log N)+I(1+T)((N-2)^2+(N-2)^2\log(N-2)^2+N+1+F)$, which is $\approx O(N^2)$ for large values of $N$.

## III. Experimental Setup

We use the following search hyper-parameters for running PASCAL: 500 individuals, maximum of 100 iterations, updating the GM based on 10% of the fittest individuals and full replacement of the population. We did not attempt to tune these parameters, and a more thorough analysis on their impact in the results is left to future research. We run PASCAL 10 times on each dataset, varying the random seed that controls evolution. In the following sections, we comment on the datasets, algorithms, and evaluation criteria that were used during the empirical analysis.

### A. Baseline Algorithms

For validating the performance of PASCAL, we compare it with 4 well-known clustering algorithms: $k$-means [2], DBSCAN [11], single and complete linkage [4], [12]. Single linkage and complete linkage are hierarchical agglomerative approaches for performing clustering. An agglomerative approach starts by generating $N$ singletons (i.e, clusters with only one object), then proceed to merge the closest clusters according to a distance measure (such as the Euclidean distance). In the process of merging clusters, a dendrogram records all merging performed in every step of the iterative process. The algorithm proceeds until all objects belong to the same cluster. Hierarchical agglomerative algorithms update the distances between clusters differently: single linkage, in particular, adopts the minimum distance between clusters, whilst complete linkage employs the maximum distance. At the end of the process, the user must specify the height at which the dendrogram will be cut, which ultimately generates a partition whose number of clusters is equal to the number of vertical edges that were cut horizontally.

$k$-means [2] is a well-known partitional algorithm that iteratively optimises the sum of squared errors (SSE) in an expectation-maximisation procedure. It starts by randomly defining $k$ prototypes, and each object is then assigned to its closest prototype. Once all objects have been assigned to a given cluster, it recomputes the prototypes based on the centroid of the cluster, and thus reassigns the objects to its updated closest centroid. This process is repeated until a maximum number of iterations is reached or until there is no variation in the cluster assignment for all objects. The value of $k$ must be specified beforehand by the user.

Finally, Density Based Spatial Clustering of Applications with Noise (or DBSCAN for short) [3] is based on the density of the neighbourhood of a given object. DBSCAN requires the user to set two parameters: a radius, which determines the maximum distance to search for neighbours, and a minimum number of neighbours to constitute a group. If an object does not have a sufficient number of neighbours in its neighbourhood, then it is considered noise.

For parametrising the baseline algorithms, we employed two different strategies. For $k$-means, single linkage and complete linkage, we executed them selecting $k$ from 2 to $\sqrt{N}$. The $k$ used for generating the partition with the best DBCV index is used as input for the algorithms. We decided to use DBCV as the validity criterion for choosing the best partition since it is the same criterion optimised by PASCAL during its evolutionary search. $k$-means was executed 10 times by varying the random seed for defining the initial prototypes for each possible value of $k$ in the interval $[2, \sqrt{N}]$.

For DBSCAN, we set a neighbourhood of 4 objects and followed the strategy proposed by Tan et al. [1] for estimating the radius parameter:

1) find the $p$-th closest neighbour ($p=4$) for each object in the dataset;
2) store the $p$-th closest neighbour distance in a unidimensional array with length $N$;
3) sort the array in crescent order;
4) take the $y$ value from the spot where the function takes the biggest leap to the next $x$ value.
5) use $y$ as radius.

The idea is that the value before the biggest leap is the maximum value which can be used before having to increase the radius too much in order to obtain different partitions. Figure 3 illustrates the unidimensional array of distances, as well as the distance before the biggest leap.

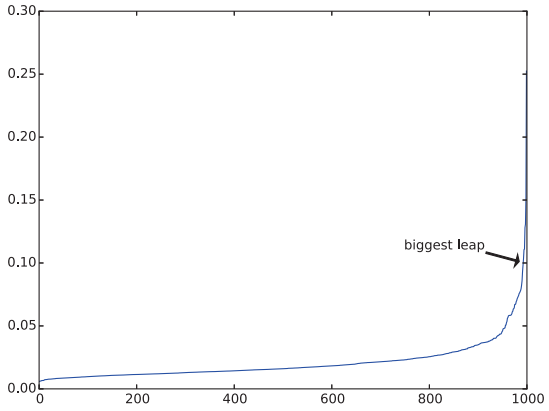


Fig. 3: Plotting of the sorted distances of the 4-th nearest neighbour for every object in the blobs2 dataset.

### B. Datasets

During the empirical analysis, we verify the performance of the algorithms in 10 datasets: 9 of them were artificially generated and one is a real-world labeled dataset. For the

real dataset, we make the further (probably naïve) assumption that the classes are equivalent to clusters. The real dataset in question is the well-known Fischer's Iris data, which contains information about petal length and width, and sepal length and width of three specimens of flowers [13]. The artificial datasets blobs2, circles0 and moons0 were generated using the Scikit Learn toolkit for the Python programming language [14]. The rest of the datasets were generated in Octave [15] using the code made available by Kools [16]. All artificial datasets are two-dimensional for the sake of visualisation. Table I presents the characteristics of all datasets, and the 9 artificial datasets can be seen in the first column of Figure 4.

TABLE I: Artificial and real datasets used in the empirical analysis.

| | dataset | # features | # objects | # clusters |
|---|---|---|---|---|
| Real | Iris | 4 | 150 | 3 |
| Artificial | blobs2 | 2 | 1,000 | 2 |
| | circles0 | 2 | 1,000 | 2 |
| | moons0 | 2 | 1,000 | 2 |
| | outlier | 2 | 1,000 | 4 |
| | clusterincluster (cinc) | 2 | 1,012 | 2 |
| | corners | 2 | 1,000 | 4 |
| | crescentfullmoon (cfm) | 2 | 1,000 | 2 |
| | halfkernel | 2 | 1,000 | 2 |
| | twospirals | 2 | 1,000 | 2 |

### C. Validity Criteria

Unlike classification, clustering is a subjective task and the quality of the result may vary according to the prior assumptions of the validity criterion that is used for evaluating the partitions. For analysing the clustering results, we decided to employ 2 clustering validity criteria: DBCV, which is the criterion optimised by PASCAL during evolution, and the Adjusted Rand Index. Note that DBCV is an *internal* validity index, which means it takes into account only the data itself to compute the partition's quality. The Adjusted Rand Index, on the other hand, is an *external* validity criterion, which compares the resulting partition with the *ground truth*, i.e., an external partition that is allegedly the expected result. Since we are using 9 artificial datasets and one labeled real-world dataset, we do have the external partitions to properly evaluate the clustering quality, and thus the *internal* validity index is just presented for the sake of completeness.

The Adjusted Rand Index (ARI) [17] analyses the conformation of a partition $Q$ to the original data distribution $R$ (ground truth). It takes into account the probability that the partition has been generated from a random distribution of objects into clusters rather than by any intelligent mechanism. The unadjusted index ranges from 0 to 1, with larger values meaning better conformations, but ARI may yield negative values if the found partition is less attractive than the random expected partition:

$$ARI = \frac{a - \frac{(a+c)(a+b)}{M}}{\frac{(a+c)+(a+b)}{2} - \frac{(a+c)(a+b)}{M}} \quad (6)$$

where:

- $a$: number of pairs of data objects from the same class in $R$ and same cluster in $Q$;
- $a$: number of pairs of data objects from the same class in $R$ and different clusters in $Q$;
- $a$: number of pairs of data objects from different classes in $R$ and to the same cluster in $Q$;
- $a$: number of pairs of data objects from different classes in $R$ and to different clusters in $Q$;
- $M = a + b + c + d$

## IV. Experimental Results

We present all results of this experimental analysis in Table II, and in Figure 4 we show the comparison between the ground truth and the partitions provided by each algorithm.

Note that PASCAL and all baseline algorithms are capable of correctly choosing the number of clusters in 8 out of the 10 datasets. Recall that we had to execute a multiple-runs procedure followed by the evaluation of an internal validity criterion to define the number of clusters for $k$-means and complete/single linkage, since they require the user to set the value of $k$. We can infer that using DBCV as an internal validity criterion for estimating the number of clusters for algorithms that need to set that parameter is indeed a good alternative. Yet, we should give emphasis to the fact that neither DBSCAN nor PASCAL require any sophisticated procedure to properly estimate the number of clusters. Moreover, note that PASCAL is, in fact, the only algorithm that does not require any procedure at all to define a set of parameters so it can be successfully executed.

The main evaluation criterion in this experimental analysis is ARI, which indicates the level of conformity between the provided partitions and the real distribution of the data. Note that both PASCAL and DBSCAN can perfectly reproduce the ground truth in 8 out of the 9 artificial datasets, substantially outperforming both hierarchical agglomerative methods and $k$-means. Given the variety of shapes present in the artificial datasets, it was expected that $k$-means would fail in reproducing the ground truth, since it is only capable of generating hyper-spherical clusters. In terms of functioning, the most similar algorithm to PASCAL is Single Linkage, but note that it fails in providing the ground truth for the outlier dataset, whereas PASCAL could reproduce it correctly.

It was also expected that PASCAL would outperform all baseline algorithms in terms of DBCV, since it is the very own criterion optimised during its evolution. Indeed, the values of DBCV for the partitions provided by PASCAL were the best for all 10 datasets. Both DBSCAN and Single Linkage had the best DBCV values in 7 datasets, substantially better than $k$-means (one dataset) and Complete Linkage (0 datasets). It seems safe to affirm that the choice of a density-based validity criterion such as DBCV proved to be a good option for looking for partitions in arbitrarily-shaped datasets, specially considering the correlation between ARI and DBCV values.

Another point worth mentioning in the experimental analysis is regarding the halfkernel dataset, which is formed by two semi-circle structures. Even though it can be hard to visualise on image, there are changes in density across the structure of each semi-circle. That is probably the reason for DBSCAN failing to detect these two semi-circles, considering that it

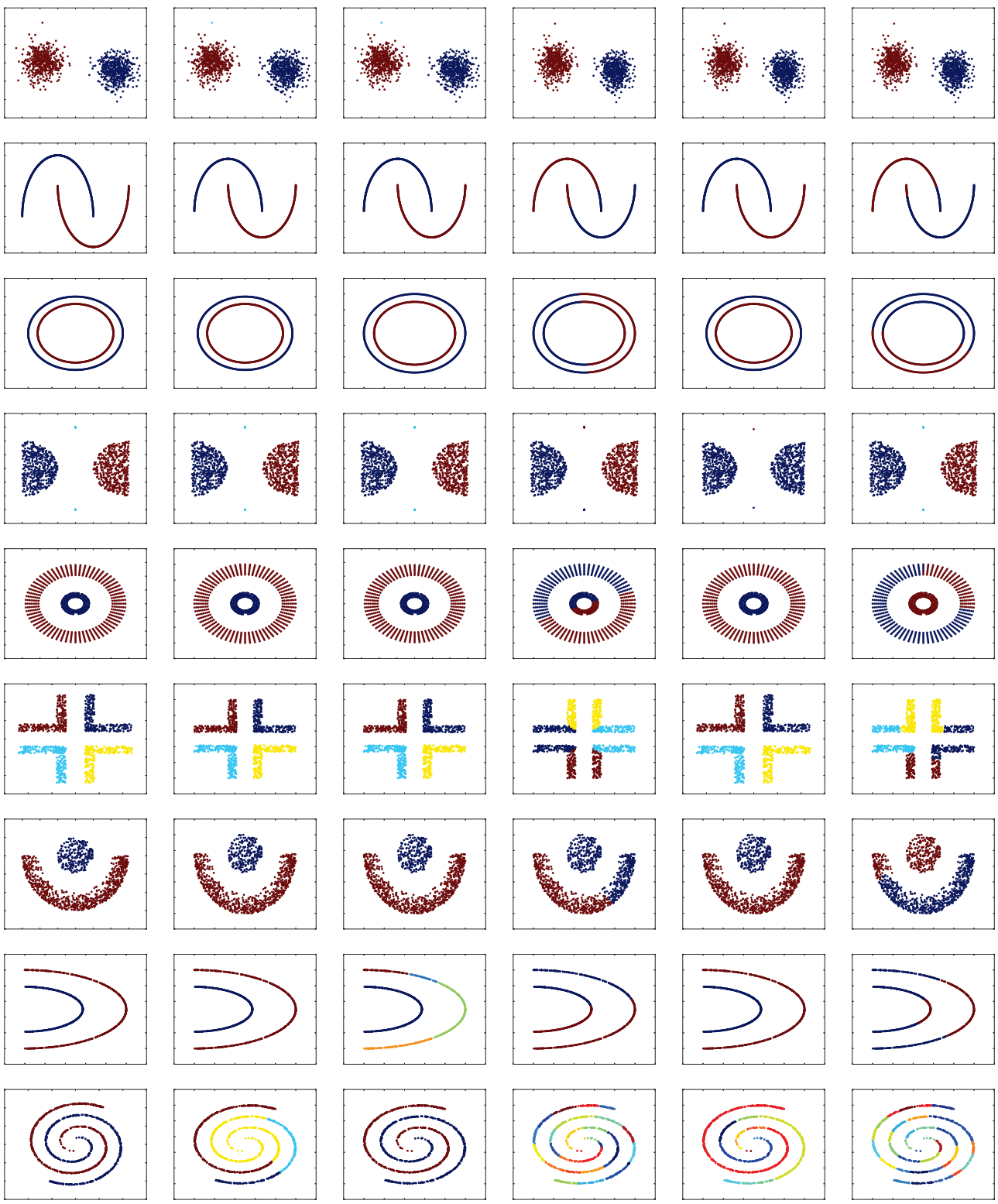


Fig. 4: Ground truth and partitions found by each algorithm. Each cluster is identified by a given colour.

TABLE II: Results for all baseline algorithms and PASCAL. Bold numbers indicate the best results for the given dataset.

| Dataset | Complete Linkage | | | Single Linkage | | | $k$-means | | | DBSCAN | | | PASCAL | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARI | DBCV | $k$ | ARI | DBCV | $k$ | ARI | DBCV | $k$ | ARI | DBCV | $k$ | ARI | DBCV | $k$ |
| blobs2 | **1.00** | -0.23 | **2** | **1.00** | -0.23 | **2** | **1.00±0.00** | -0.23±0.00 | **2±0** | **1.00** | **-0.09** | **2** | **1.00±0.00** | **-0.09±0.00** | **2±0** |
| moons0 | 0.59 | -0.81 | **2** | **1.00** | **0.74** | **2** | 0.53±0.00 | -0.82±0.00 | **2±0** | **1.00** | **0.74** | **2** | **1.00±0.00** | **0.74±0.00** | **2±0** |
| circles0 | 0.00 | -0.39 | **2** | **1.00** | **0.30** | **2** | 0.00±0.00 | -0.40±0.00 | **2±0** | **1.00** | **0.30** | **2** | **1.00±0.00** | **0.30±0.00** | **2±0** |
| outlier | **1.00** | 0.09 | **2** | 0.00 | -0.33 | **2** | 0.99±0.00 | -0.37±0.00 | **2±0** | **1.00** | **0.54** | **2** | **1.00±0.00** | **0.54±0.00** | **2±0** |
| cinc | 0.53 | -0.25 | **2** | **1.00** | **0.38** | **2** | 0.00±0.00 | -0.93±0.02 | **2±0** | **1.00** | **0.38** | **2** | **1.00±0.00** | **0.38±0.00** | **2±0** |
| corners | 0.37 | -0.73 | **4** | **1.00** | **0.28** | **4** | 0.70±0.30 | -0.23±0.55 | **4±0** | **1.00** | **0.28** | **4** | **1.00±0.00** | **0.28±0.00** | **4±0** |
| cfm | 0.65 | -0.75 | **2** | **1.00** | **0.03** | **2** | 0.21±0.01 | -0.67±0.04 | **2±0** | **1.00** | **0.03** | **2** | **1.00±0.00** | **0.03±0.00** | **2±0** |
| halfkernel | 0.02 | -0.97 | **2** | **1.00** | **0.35** | **2** | 0.09±0.13 | -0.88±0.00 | **2±0** | 0.68 | 0.13 | 5 | **1.00±0.00** | **0.35±0.00** | **2±0** |
| twospirals | 0.03 | -0.58 | 24 | 0.31 | -0.33 | 24 | 0.05±0.00 | -0.47±0.04 | 24±0 | **1.00** | -0.57 | **2** | 0.46±0.06 | **-0.25±0.04** | 5±1 |
| iris | 0.22 | -0.65 | 2 | **0.57** | **0.22** | 2 | **0.57±0.00** | **0.22±0.00** | 2±0 | **0.57** | -0.05 | 2 | **0.57±0.00** | **0.22±0.00** | 2±0 |
| Victories | 2 | 0 | 8 | 8 | 7 | 8 | 2 | 1 | 8 | 9 | 7 | 8 | 9 | 10 | 8 |

detects the lower-density regions as inter-cluster areas, and thus ends up generating more clusters than necessary. PASCAL, on the other hand, is perfectly capable of detecting the two clusters thus yielding the best ARI value.

Notwithstanding, PASCAL did fail to properly detect the two spirals in the twospirals dataset, whereas DBSCAN was the only algorithm to correctly detect the groups. More interestingly, the twospirals dataset was the only one in which PASCAL had a variety of behaviour in its 10 runs, finding partitions ranging from 4 to 6 clusters, instead of the real value of 2. We believe that happened because of a particularity with the DBCV criterion, as follows. The centre of the dataset is a low-density area with objects from the two spirals. By following the course of the spirals, the objects condense into a high-density distribution, misleading DBCV to understand that clustering the whole centre is a good idea. Indeed, PASCAL achieves the largest DBCV value for this dataset, clearly indicating that DBCV is not particularly suited for this problem. We are already studying new strategies for modifying DBCV so it can cope with this scenario while keeping the good results achieved so far.

Finally, it is worth mentioning that PASCAL is quite stable across multiple runs. Only for the twospirals dataset, which seems to deceive the behaviour of DBCV, PASCAL ended up generating different results. Perhaps this is a particular case in which tuning the search parameters of the EDA would yield more interesting results. For instance, we assume that perhaps with a larger number of individuals (and perhaps a larger budget) PASCAL would be capable of identifying more interesting conformations that could lead to larger DBCV values. Nevertheless, since the whole point of PASCAL is to be *parameterless*, suggesting to tune the search parameters may not be the right solution for the problem, so we prefer to investigate modifications of DBCV or perhaps the inclusion of multiple validity criterion within the fitness function so the final user does not need to worry with setting parameters for PASCAL at all.

## V. Related Work

Using minimum spanning trees for clustering is not a novel approach. For instance, the dendrogram produced by single linkage (introduced in Section III-A) is in fact a MST. The difference between our approach to single linkage is two-fold. First, PASCAL does not require setting the value of $k$ (or, in other words, a horizontal cut in the dendrogram) in order to assign objects to clusters. Second, PASCAL allows the "regretting" from a cluster assignment for a given object. In the dendrogram, there is only one partition which yields $k$ clusters. In order to obtain different partitions for the same number of groups, the user must run other hierarchical agglomerative clustering algorithms such as complete linkage. PASCAL, on the other hand, has $n!/(n-k)!$ possible partitions for each value of $k$ ranging in $[2, N-1]$.

The work of Zhou et al. [18] propose two procedural algorithms for performing clustering with minimum spanning trees, one $k$-constrained and the other unconstrained. Since PASCAL automatically detects the number of clusters, we describe here the unconstrained algorithm. It starts by building a MST from the dataset, and then it iteratively removes edges from the MST. By removing an edge, it produces two clusters of data objects. The approach used for removing edges is to remove those that contribute the most for increasing the weighted standard deviation of all edges in the set of subtrees. It then performs a 6-th order regression analysis with the information of how much is reduced in terms of standard deviation with the number of removed edges. When removing an edge ceases to decrease the standard deviation of the partition, the corresponding value of $k$ is chosen and the subtrees generated from that configuration is returned.

Regarding evolutionary algorithms for clustering, Alves et al. [8], [19] propose a Fast Evolutionary Algorithm for Clustering (F-EAC), a mutation-based algorithm which further improves the EAC [20]. F-EAC encodes all $N$ objects of the dataset in one array of $N$ positions. For each position, it randomly assigns a value in the range $[1, k]$ during the first generation, where $k$ is the number of partitions. The initial value of $k$ is a starting point, since it is constantly changed by the algorithm within its procedure. F-EAC proceeds to mutate individuals in order to update cluster assignments for each object, either by splitting or merging clusters. Two functions are used for fitness evaluation across the several variations of F-EAC presented in the work: Simplified Silhouette Width Criterion (SSWC), which is a simplification over the original silhouette; and Rand Index, the original unadjusted version. The authors develop a set of F-EAC variants, which can accurately predict the correct number of clusters and are faster than the original EAC implementation. As mentioned before, F-EAC requires an initial value of $k$ to initiate the clustering process. Furthermore, since it uses the SSWC as fitness function, it tends to favour spherical clusters. The variation in which it uses the Rand Index is not realistic since

in most domains there is no known ground truth.

We perform clustering with an Estimation of Distribution Algorithms in our previous work [21], in which we present Clus-EDA. Similarly to the work of Alves et al. [19], it also encodes all $N$ objects of the dataset in an unidimensional array. However, since it is an EDA, it samples novel individuals from an univariate marginal distribution. The GM is implemented as an unidimensional array, with each position representing an object and each value a probability of that object to become a medoid. Non-medoid objects are assigned to their closest medoid. All values in the GM are initialised in $\frac{k^*}{N}$, with $k^*$ being an estimate of the value of $k$ provided by a heuristic that involves $k$-means and an internal validity criterion. Novel individuals are sampled from the GM, which is in turn updated using the fittest individuals, evaluated by the Simplified Silhouette Width Criterion. Clus-EDA is compared with F-EAC, $k$-means and UPGMA, an hierarchical agglomerative algorithm with regard to the following criteria: Davies-Bouldin, Adjusted Rand Index, and the Silhouette Width Criterion. Clus-EDA is capable of properly adjusting the initial value of $k$ into the correct number of clusters, whilst also providing the best Davies-Bouldin and SWC index for 7 out of 9 datasets that were analysed. Nevertheless, similarly to F-EAC, Clus-EDA also uses SSWC as its fitness function, thus favouring spherical groups and being unable to have a good performance in arbitrarily shaped datasets, besides needing an initial value of $k$ to work from.

## VI. Conclusions and Future Work

In this paper, we presented PASCAL, a novel and effective EDA for performing data clustering that is capable of addressing arbitrarily-shaped clusters without the need of setting specific and decisive parameters such as the number of clusters, minimum density, radius, etc. PASCAL makes use of a MST to identify possible constraints of must-link/cannot link between pairs of objects, and it optimises a density-based validity criterion during its search for the best partition.

PASCAL is compared to well-known clustering algorithms that employ different strategies to perform clustering, such as $k$-means [2], Single Linkage [12], Complete Linkage [12], and DBSCAN [11]. By performing an empirical analysis with 10 datasets whose ground truth partitions were previously known, we show that PASCAL is capable of not only correctly identifying the number of clusters but also of presenting the largest possible conformation between the predicted partitions and the real ones in 8 out of the 10 datasets. Indeed, PASCAL seems to perform as strongly as DBSCAN, though with the further advantage of not requiring any critical clustering parameters, whereas DBSCAN requires two of them. As future work, we aim to investigate which modifications are required in PASCAL's density-based fitness function so it can successfully deal with scenarios that PASCAL failed to identify the ground truth. Moreover, we intend to verify whether a multi-objective fitness function would be well-suited for addressing this issue.

## Acknowledgements

The authors would like to gratefully acknowledge Motorola Mobility for funding this research.

## References

[1] P.-N. Tan, M. Steinbach, and V. Kumar, "Introduction to data mining," 2005.

[2] S. Lloyd, "Least squares quantization in pcm," *IEEE Trans. Inf. Theor.*, vol. 28, no. 2, pp. 129–137, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1982.1056489

[3] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[4] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[5] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. De Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Trans. Sys. Man Cyber Part C*, vol. 39, no. 2, pp. 133–155, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1109/TSMCC.2008.2007252

[6] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed. MIT press, 2009.

[8] E. R. Hruschka, R. J. G. B. Campello, and L. N. de Castro, "Evolving clusters in gene-expression data," *Information Sciences*, vol. 176, no. 13, pp. 1898–1927, Jul. 2006.

[9] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[10] D. Moulavi, P. A. Jaskowiak, R. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA*, 2014.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[12] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[13] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] S. H. John W. Eaton, David Bateman and R. Wehbring, *GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations*, 2015. [Online]. Available: http://www.gnu.org/software/octave/doc/interpreter

[16] J. Kools, *6 Functions for generating artificial datasets*, accessed: 2016-01-12. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generating-artificial-datasets

[17] L. Vendramin, R. J. Campello, and E. R. Hruschka, "Relative clustering validity criteria: A comparative overview," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 3, no. 4, pp. 209–235, 2010.

[18] Y. Zhou, O. Grygorash, and T. F. Hain, "Clustering with minimum spanning trees," *International Journal on Artificial Intelligence Tools*, vol. 20, no. 01, pp. 139–177, 2011.

[19] V. S. Alves, R. J. Campello, and E. R. Hruschka, "Towards a fast evolutionary algorithm for clustering," in *IEEE Congress on Evolutionary Computation (IEEE CEC 2006)*. IEEE, 2006, pp. 1776–1783.

[20] E. R. Hruschka, L. N. de Castro, and R. J. Campello, "Evolutionary algorithms for clustering gene-expression data," in *IEEE 13th International Conference on Data Mining*. IEEE, 2004, pp. 403–406.

[21] H. E. L. Cagnini, R. C. Barros, C. V. Quevedo, and M. P. Basgalupp, "Medoid-based data clustering with estimation of distribution algorithms," in *ACM Symposium on Applied Computing (SAC 2016)*, 2016.