Subspace Clustering Using Evolvable Genome Structure

Sergio Peignier
Université de Lyon
INSA-Lyon, CNRS, INRIA
LIRIS, UMR5205
F-69621, France
Sergio.Peignier@inria.fr

Christophe Rigotti
Université de Lyon
INSA-Lyon, CNRS, INRIA
LIRIS, UMR5205
F-69621, France
Christophe.Rigotti@insa-lyon.fr

Guillaume Beslon
Université de Lyon
INSA-Lyon, CNRS, INRIA
LIRIS, UMR5205
F-69621, France
Guillaume.Beslon@inria.fr

ABSTRACT

In this paper we present an evolutionary algorithm to tackle the subspace clustering problem. Subspace clustering is recognized as more difficult than standard clustering since it requires to identify not only the clusters but also the various subspaces where the clusters hold. We propose to tackle this problem with a bio-inspired algorithm that includes many bio-like features like variable genome length and organization, functional and non-functional elements, and variation operators including chromosomal rearrangements. These features give the algorithm a large degree of freedom to achieve subspace clustering with satisfying results on a reference benchmark with respect to state of the art methods. One of the main advantages of the approach is that it needs only one subspace clustering ad-hoc parameter: the maximal number of clusters. This is a single and intuitive parameter that sets the maximal level of details of the clustering, while other algorithms require more complicated parameter space exploration. The other parameters of the algorithm are related to the evolution strategy (population size, mutation rate, ...) and for them we use a single setting that turns out to be effective on all the datasets of the benchmark.

1. INTRODUCTION

Subspace clustering is recognized as more general and more difficult than clustering. A subspace clustering task searches for objects sharing similar feature values, and also at the same time searches for the subspaces where these similarities appear, while usual clustering only looks for groups of objects similar over the whole data space. Subspace clustering can be thought as *"similarity examined under different representations"* [Patrikainen and Meila, 2006] and is particularly useful when dealing with high dimensional spaces [Kriegel et al., 2009].

According to [Banzhaf et al., 2006], bio-inspired optimization algorithms could be improved by incorporating knowledge from molecular and evolutionary biology. A promising source of advances in optimization is one of the important phenomena in evolutionary biology: the dynamic evolution of the genome structure. Several studies showed for instance that an evolvable genome structure allows evolution to modify the effects that evolution principles (e.g., mutations) have on individuals, a phenomenon known as *evolution of evolution* [Hindré et al., 2012]. Evolvable genome structure can encompass various aspects, such as variable genome length or variable ratio of functional versus non-functional elements [Knibbe et al., 2007]. Among the state-of-the-art formalisms used for *in silico* experimental evolution and reviewed in [Hindré et al., 2012], two models enable genome structure evolution: [Knibbe et al., 2007] and [Crombach and Hogeweg, 2007], and have inspired key aspects of our work.

In this paper, we present Chameleoclust, an evolutionary subspace clustering algorithm that incorporates a genome having an evolvable structure. The genome is a *coarse-grained* genome, inspired on [Crombach and Hogeweg, 2007], and defined as a list of tuples of numbers. These tuples are mapped at the phenotype level to denote core point locations in different dimensions, which are then used to build the subspace clusters. In Chameleoclust, the genome also contains a variable proportion of non-functional elements as in [Knibbe et al., 2007], and is subject to local mutations and to large random rearrangements similar to those used in [Knibbe et al., 2007] and [Crombach and Hogeweg, 2007], namely: large deletions, duplications and translocations. The local mutations and rearrangements modify the genome elements but also the genome length and the proportion of non-functional elements. The key intuition in the design of the Chameleoclust algorithm is to take advantage of such an evolvable structure to detect various numbers of clusters in subspaces of various dimensions.

Chameleoclust is assessed using a reference framework for subspace clustering evaluation[Müller et al., 2009], and compared to state-of-the-art algorithms on both real and synthetic datasets. The experiments show that Chameleoclust obtains competitive results. Moreover, these results can be achieved with a single parameter related to the domain, i.e., the maximal number of clusters.

The rest of the paper is organized as follows. The next section introduces the proposed algorithm, and Sections 3 and 4 describe respectively the evaluation method and results. Section 5 presents the related work and we conclude with a summary in Section 6.

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754709

2. Chameleoclust

Chameleoclust includes many bio-like features such as a variable genome length and organization, presence of both functional and non-functional elements, and variation operators including large chromosomal rearrangements. These features, inspired by the *in silico* experimental evolution formalisms of [Knibbe et al., 2007] and [Crombach and Hogeweg, 2007], give the algorithm a large degree of freedom by making the genome structure evolvable. Chameleoclust takes advantage of this structural flexibility to build subspace clustering with various number of clusters and in subspaces having different numbers of dimensions.

Dataset and clusters.

A dataset S is a set of objects described in $\mathbb{R}^D$ by D features (the coordinates of the objects). The size of S is the number of objects in S, and D is the number of dimensions (i.e., the dimensionality) of S. Each dimension is defined by a number from 1 to D and the set of all dimensions of the dataset is denoted $\mathcal{D} = \{1, \ldots, D\}$. The algorithm takes as input a dataset S and a parameter c_{max} that is the maximal number of desired clusters. The algorithm outputs a subspace clustering in the form of a set of clusters, where each cluster is defined by a set of objects and a set of dimensions.

Overall clustering principle.

Each individual encodes in its genome a subspace clustering. More precisely a genome defines a set of so called *core points* located in various subspaces having possibly less than D dimensions. If the objects of the dataset tends to form groups around these core points, then a high fitness is associated to the corresponding individual. The reproduction (including selection and mutations) is performed for a whole generation in a synchronized way. After a given number of generations the process stops and the subspace clustering corresponding to the individual having the highest fitness is retained.

Preprocessing.

As in many typical clustering problems, the first step is to standardize the dataset to ensure that all features could have similar impact on the distance computation during the clustering. Thus each feature value x is replaced by its z-score: $z = \frac{x-\mu}{\sigma}$, where μ is the dataset mean and σ is dataset standard deviation for the given feature. After standardization, data values in different dimensions are independent of the offset and scale, and features have all the same zero mean and the same unitary dispersion. Finally the maximal value among all absolute values of the z-score of all features is computed and is noted x_{max} in the rest of the paper.

Genome structure.

A genome Γ is a list $[\gamma_1, \ldots, \gamma_i, \ldots, \gamma_n]$ of tuples of the form $\gamma_i = \langle g_i, c_i, d_i, x_i \rangle$, where $g_i \in \{0, 1\}$ indicates if γ_i is a functional element of the genome ($g_i = 1$) or not ($g_i = 0$), and c_i, d_i, x_i will be used to define the individual phenotype and have the following specific domains: $c \in \{1, \ldots, c_{max}\}$, $d \in \{1, \ldots, D\}$ and $x \in ValCoord$, with $ValCoord = \{j \times x_{max}/1000 \mid j \in \{-1000, \ldots, 1000\}\}$, i.e. all values from $-x_{max}$ to x_{max} with step $x_{max}/1000$.

Phenotype.

A phenotype is simply a set of core points. Informally a core point is a specific point, around which objects can be grouped to form a subspace cluster. The number of core points cannot exceed the maximal number of desired clusters c_{max}. Each core point is identified by a number $c \in [1, c_{max}]$ and is denoted p_c. The intuition of the genotype-phenotype mapping is that each functional element of the genome $\langle 1, c, d, x \rangle$ is a contribution of value x to the location of core point p_c in dimension d. More precisely, let x_d be the coordinate of p_c for dimension d, then x_d is the sum of all the values x contained in a tuple of the form $\langle 1, c, d, x \rangle$ in the genomeΓ . The subspace associated to p_c (and for wich p_c is defined) is the set of dimensions $\mathcal{D}_{p_c} = \{d \mid \langle 1, c, d, x \rangle \in \Gamma \text{ for some } x\}$, i.e., the dimensions that contribute to p_c inΓ . Notice that the non-functional elements of Γ do not contribute to the phenotype.

For a given dataset $\mathcal{S}$, a phenotype Φ defines a subspace clustering of $\mathcal{S}$, by associating each object of $\mathcal{S}$ to the best matching core point in Φ. A non empty set of objects associated to a core point p_c forms a cluster in subspace $\mathcal{D}_{p_c}$. The precise definition of the notion of *best match* is given in paragraph *Fitness* here after.

Notice that the length of the genome can be different among individuals, leading to phenotypes containing different numbers of core points in various subspaces and thus defining subspace clustering models with different number of clusters in subspaces having different number of dimensions.

Mutation operators.

Each new genome is copied from a parent and modified by biologically inspired mutation operators of two kinds: Global rearrangements and point mutations. These operators are general mutation operators and are not guided by some criteria related to the subspace-clustering task. The model uses a single point mutation operator defined as follows for a genomeΓ.

- **Point substitution**: Let $\gamma_i \in \Gamma$ and $k \in \{1, 2, 3, 4\}$, both uniformly chosen, the k-th element of the tuple γ_i is replaced by a new random number drawn uniformly in its associated range:

$$\gamma_i \leftarrow \begin{cases} \langle \mathcal{U}(\{0,1\}), c, d, x \rangle & \textbf{if } k = 1 \\ \langle g, \mathcal{U}(\{1, \ldots, c_{max}\}), d, x \rangle & \textbf{if } k = 2 \\ \langle g, c, \mathcal{U}(\{1, \ldots, D\}), x \rangle & \textbf{if } k = 3 \\ \langle g, c, d, \mathcal{U}(ValCoord) \rangle & \textbf{if } k = 4 \end{cases}$$

  where $\mathcal{U}$ denotes the uniform random selection of a element in a set.

For the rearrangements, Γ is considered as being circular (as bacterial genomes). This means that the tuple γ_n is adjacent to the tuple γ_1. In order to define the possible rearrangements let us define three basic operators.

- Sublist extraction operator:

$$[\gamma_1, \ldots, \gamma_n]_{i,j} = \begin{cases} [\gamma_i, \ldots, \gamma_j] & \textbf{if } i < j \\ [\,] \text{ (the empty list)} & \textbf{if } i \geq j. \end{cases}$$

- List concatenation operator:

$$[\gamma_1, \ldots, \gamma_n] + [\gamma'_1, \ldots, \gamma'_m] = [\gamma_1, \ldots, \gamma_n, \gamma'_1, \ldots, \gamma'_m]$$

- Tuple merge operator M: Let $\gamma = \langle g, c, d, x \rangle$ and $\gamma' = \langle g', c', d', x' \rangle$, then four merges are defined depending on a merge index k:

$$M(\gamma, \gamma', k) = \begin{cases} \langle g, c', d', x' \rangle & \textbf{if } k = 1 \\ \langle g, c, d', x' \rangle & \textbf{if } k = 2 \\ \langle g, c, d, x' \rangle & \textbf{if } k = 3 \\ \langle g, c, d, x \rangle & \textbf{if } k = 4 \end{cases}$$

Rearrangements are responsible for changing the tuples positions, increasing or decreasing the genome length. In addition rearrangement breakpoints operate inside tuples, breaking them and recombining them according to the rearrangement operation, creating new tuples from old genetic material. The model uses three kinds of rearrangements: large deletions, large duplications and large translocations. For a rearrangement of a genome $\Gamma = [\gamma_1, \dots, \gamma_n]$, a portion of$\Gamma$ bounded by two tuples $\gamma_i, \gamma_j \in \Gamma$ is considered, where i and j are uniformly chosen in $\{1, \dots, n\}$. A breakpoint index is also uniformly chosen in $\{1, \dots, 4\}$ to specify where the rearrangement limit is located within the bounding tuples. The three rearrangement operators can then be defined as follows:

- **Large deletions**: The segment between tuples γ_i and γ_j is excised.

  If $i \leq j$:

  $\Gamma \leftarrow \Gamma_{1,i-1} + M(\gamma_i, \gamma_j, k) + \Gamma_{j+1,n}$

  If $i > j$, because of genome circularity, we have:

  $\Gamma \leftarrow \Gamma_{j+1,i-1} + M(\gamma_i, \gamma_j, k)$

- **Large duplications** : The segment between tuples γ_i and γ_j is copied and inserted at the location of a third tuple γ_p (uniformly chosen).

  If $i \leq j$:

  $\Gamma \leftarrow \Gamma_{1,p-1} + M(\gamma_p, \gamma_i, k) + \Gamma_{i+1,j-1} + M(\gamma_j, \gamma_p, k) + \Gamma_{p+1,n}$

  If $i > j$, because of genome circularity, we have:

  $\Gamma \leftarrow \Gamma_{1,p-1} + M(\gamma_p, \gamma_j, k) + \Gamma_{j+1,n} + \Gamma_{1,i-1} + M(\gamma_i, \gamma_p, k) + \Gamma_{p+1,n}$

- **Large translocations**: The segment between tuples γ_i and γ_j is cut and inserted at the location of a third tuple γ_p (uniformly chosen) such that $p \notin [i, j]$ if $i \leq j$ and $p \notin [1, j] \cup [i, n]$ if $i > j$. Translocation can be defined with the two previous rearrangement operations. At first the segment between tuples γ_i and γ_j is duplicated at the location of the third tuple γ_p, and then it is deleted.

During the reproduction of an individual, the mutations stage is defined as follows. For each of the three kinds of rearrangement operations, the number of rearrangements per genome is drawn from a binomial law $\mathcal{B}(L, u_m)$ where L is the genome size and u_m is the mutation rate (same rate for all mutation operators). The rearrangements of the three kinds are then performed in a random order. Once rearrangements have been applied, the number of point substitutions per genome is drawn from a binomial law $\mathcal{B}(L', u_m)$ where L' is the genome size after applying the rearrangement operations, and then all these point substitutions are carried out.

Fitness.

For a given dataset $\mathcal{S}$, the fitness of a individual of phenotype Φ is related to the quality of the subspace clustering defined by Φ over $\mathcal{S}$. This quality measure is an internal measure based on a distance, and reflecting how the objects in $\mathcal{S}$ tend to form groups around core points ofΦ.

In [Beyer et al., 1999] and [Aggarwal et al., 2001] it has been shown that distance comparisons are less meaningful when dimensionality increases, this effect is called the *concentration effect* of the distances. Furthermore, distances do not have the same meaning in subspaces with different numbers of dimensions: It is not fair to compare distances calculated in subspaces with different dimensionality.

It has been shown in [Aggarwal et al., 2001] that the Manhattan distance is robust to the concentration effect. In the Chameleoclust algorithm, the distance used is the *Manhattan segmental distance* introduced in [Aggarwal et al., 1999] for the well known subspace clustering algorithm PROCLUS. It is a normalized version of the classic Manhattan distance to compare distances in subspaces with different number of dimensions. Let y_1 and y_2 be two points in a space over the set of dimension $\mathcal{D}$, and $y_{1,i}$ (resp. $y_{2,i}$) denotes the coordinate of y_1 (resp. y_2) in the dimension i of $\mathcal{D}$. Then, the Manhattan segmental distance is defined as:

$$d_{\mathcal{D}}(y_1, y_2) = \sum_{i \in \mathcal{D}} \frac{|y_{1,i} - y_{2,i}|}{|\mathcal{D}|}$$

The functions to evaluate the subspace clustering models corresponding to phenotypes and the fitness of the individuals are defined as follows.

- A function $\mathcal{E}(x, p_c)$ is used to assess the mismatch of the assignment of an object $x \in \mathcal{S}$ in space $\mathcal{D}$ to a core point p_c in subspace $\mathcal{D}_{p_c}$. The highest is $\mathcal{E}(x, p_c)$, the worst is the association of x to p_c. This function is defined by:

  $$\mathcal{E}(x, p_c) = \frac{|\mathcal{D}_{p_c}| \cdot d_{\mathcal{D}_{p_c}}(x, p_c) + |\mathcal{D} \setminus \mathcal{D}_{p_c}| \cdot d_{\mathcal{D} \setminus \mathcal{D}_{p_c}}(x, \mathcal{O})}{\mid \mathcal{D} \mid}$$

  where $\mathcal{O}$ is the centroid of dataset $\mathcal{S}$, i.e., the origin of the coordinate system after the z-score standardization. The mismatch evaluation $\mathcal{E}(x, p_c)$ increases with the distance between the core point p_c and the object x (i.e., $d_{\mathcal{D}}(x, p_c)$). $\mathcal{E}(x, p_c)$ also increases if the subspace $\mathcal{D}_{p_c}$ has not enough dimensions to explain the shift of x with respect to $\mathcal{O}$ (i.e., $d_{\mathcal{D}' \setminus \mathcal{D}}(x, \mathcal{O})$).

- Each object $x \in \mathcal{S}$ is assigned to the core point $p_c \in \Phi$ for which $\mathcal{E}(x, p_c)$ is minimal (in the rare cases where several core points lead to the same minimal value, then one of them is chosen nondeterministically). Let $\mathcal{S}_{p_c}$ be the set of objects associated to p_c, then if $\mathcal{S}_{p_c}$ is not empty, the core point p_c defines the subspace cluster $\langle \mathcal{S}_{p_c}, \mathcal{D}_{p_c} \rangle$.

- Finally the fitness $\mathcal{F}$ can be defined as the opposite of the sum of the mismatches of the best possible assignments of the objects:

  $$\mathcal{F} = - \sum_{p_c \in \Phi} \sum_{x \in \mathcal{S}_{p_c}} \mathcal{E}(x, p_c)$$

The fitness function $\mathcal{F}$ goes to 0 when the evaluation of the mismatches between objects and core points tends to 0 (perfect match), and is strongly negative when objects and core points are poorly related. Notice that a core point p_c with no associated object ($\mathcal{S}_{p_c} = \emptyset$) is not penalized, and its corresponding functional elements in the genome may then be preserved for further exploration.

Population.

Each individual can be perceived as an asexual artificial organism containing a single chromosome. The population evolves during T generations. At each generation the population is completely renewed but its size N remains constant over time. The following rank based selection method is used for reproduction. The individual of the current generation are ranked according to their fitness, in increasing order of performance (the worst has rank 0 and the best rank N). Then for each of the N individuals of the offspring generation, the parent of this individual is determined by a trial over a N classes multinomial law, where each class is associated to an individual of the current generation. For this multinomial law, a class associated to an individual α has a success probability $p_\alpha = (s-1)\frac{s^{N-r_\alpha}}{s^N-1}$ where r_α is the rank of the individual α and s the selection pressure parameter. In order to avoid the best fitness to decrease Chameleoclust uses an elitist selection method. More precisely, it always adds in the next generation an unchanged copy of the best current individual, and performs the random reproduction using only $N-1$ trials.

For the first generation, every initial individual is generated with a genome of size 100 containing random tuples of the form: $\langle g, \mathcal{U}(\{1, c_{max}\}), \mathcal{U}(\{1, D\}), \mathcal{U}(ValCoord\})\rangle$, with $g = 1$ for 1/3 of them (functional elements) and $g = 0$ for the others (non-functional elements).

3. EXPERIMENTAL SETUP

Experimental protocol.

In order to evaluate and compare Chameleoclust to state-of-the-art algorithms, we used the evaluation framework of reference designed for subspace clustering and described in [Müller et al., 2009]. We clustered with Chameleoclust the same datasets and computed the same evaluation measures as recommended by [Müller et al., 2009]. We report in the Table 2 and Figure 3 the results given in [Müller et al., 2009] for the state-of-the-art algorithms, together with the results obtained for Chameleoclust.

In the framework of [Müller et al., 2009], as each algorithm requires several parameters (from 2 to 9), they are executed with many different parameter settings to explore the parameter space. Then, using an external labeling of the objects, only the subspace clusterings that are among the best (with respect to the external labeling) are retained. So, the results reported for these algorithms are in some sense the best possible subspace clusterings that could be achieved if we were able to find the most appropriated parameter values. Since generally no external labeling is available when we search for clusters, parameter tuning is most of the time a difficult task and these high quality subspace clusterings are likely to be hard to obtain.

An important point to notice, is that for Chameleoclust we did not perform any parameter optimization using external information. We simply ran Chameleoclust and took the subspace clustering defined by the individual of the last generation having the best fitness. Since the algorithm is non-deterministic, we ran it 10 times in the same conditions and report the minimal, maximal and mean values of the measures over these 10 runs. So, we compare clusterings effectively found by Chameleoclust to the best clusterings that could potentially be found by the other algorithms. All experiments were run on a quad-core Intel 2.67GHz CPU running Linux Ubuntu 14.04, and using a single core.

Datasets.

We studied Chameleoclust performances on real word data using the six benchmark datasets selected in [Müller et al., 2009] for their representativity: *breast, diabetes, liver, glass, shape, pendigits* and *vowel* (most of them coming from the UCI archive [Bache and Lichman, 2013]). These datasets have different dimensionalities and contain different numbers of objects. These objects are already structured in classes, and the class membership is used by quality measures to assess the cluster *purity*. However the number of classes does not necessarily reflect the number of subspace clusters, since even within a class the objects can form several clusters in different subspaces.

We also ran Chameleoclust on the 16 synthetic benchmark datasets provided by [Müller et al., 2009]. These datasets are particularly useful to study the algorithm performances, as the true clusters and their subspaces are known. They have different number of dimensions (5, 10, 15, 20, 25, 50 and 75), contain different number of objects (1500, 2500, 3500, 4500 and 5500), and various percentages of added noise (0, 10, 30, 50 and 70%). Each dataset contains 10 hidden subspace clusters.

For the real datasets we ran Chameleoclust using all objects, while for the synthetic dataset it was executed on random samples containing 20% of the objects to reduce the duration of the experiments. However, in both cases we used 100% of the objects for the evaluation of the clustering quality. This means that for a synthetic dataset after having computed a subspace clustering with 20% of the objects, then we associated 100% of the objects to the clusters (using the best matching core points). All datasets and additional material are made available by the authors of [Müller et al., 2009] at http://dme.rwth-aachen.de/openSubspace/evaluation.

Parameter setting.

We used the same setting of the evolution strategy parameters for **all** datasets: mutation rate $u_m = 0.005$, selection pressure $s = 0.8$, population size $N = 100$, and number of generations $T = 5000$. With 5000 generations the algorithm achieved a good convergence for fitness and genome length, as illustrated in Figure 1 and Figure 2. A single parameter needed to be changed: c_{max} the maximal number of subspace clusters that are built. However, this parameter does not require a fine tuning since Chameleoclust adapts the number of subspace clusters between 1 and c_{max}. It was set to 20 for all synthetic datasets and for three real datasets: *shape*, *pendigits* and *breast*. For the other real datasets, Chameleoclust was also tried with $c_{max} = 20$, but in this case the algorithm output exactly 20 subspace clusters. This meant that it did not succeed to regulate the number of subspace clusters because c_{max} was to small. Thus for these datasets the clusterings were repeated with $c_{max} = 100$ that turned out then to be a sufficient upper bound.

Evaluation measures.

In order to compare our algorithm to the others, we used the same standard evaluation measures for clusters and subspace clusters as [Müller et al., 2009]: entropy, accuracy, F1, RNIA and CE. We performed also the same simple transformation of entropy and RNIA, by computing $\overline{RNIA} = 1 - RNIA$ and $\overline{entropy} = 1 - entropy$ to have all evaluation measures ranging from 0 (low quality) to 1 (high quality). The three first measures (entropy, accuracy and F1) reflect how well objects that should have been grouped together were effectively grouped. The two last measures (RNIA and CE) take into account the way the objects are grouped and also relevancy of the subspaces found by the algorithm. For these measures, when the *true* dimensions of the subspace clusters are not known (for real datasets), then as in [Müller et al., 2009] all dimensions have been considered as relevant. We refer the reader to [Müller et al., 2009] for a detailed presentation of the evaluation measures.

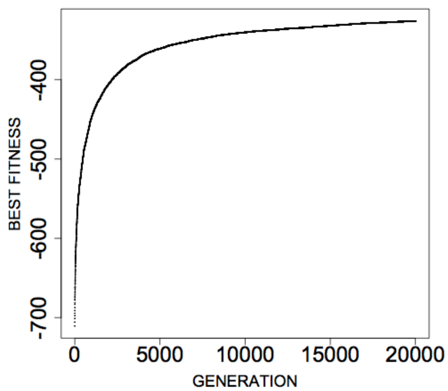

Figure 1: Average of the best fitness for 10 runs on one of the synthetic dataset (1500 objects, 10 dimensions, 0% noise).

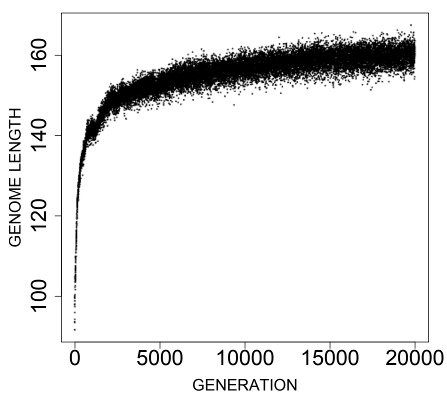

Figure 2: Average genome length of all individuals over 10 runs on one of the synthetic dataset (1500 objects, 10 dimensions, 0% noise).

4. EXPERIMENTAL RESULTS

Real dataset.

As aforementioned, we computed the minimum, the maximum and the mean of the evaluation measures over 10 standard runs of Chameleoclust, while for the other algorithms the measure values are taken from the best possible runs (the ones leading to the highest evaluation measures over the parameter space). Even though Chameleoclust has been executed on a computer (2.67GHz CPU) different from the one used by [Müller et al., 2009] (2.3GHz CPU), we report the runtimes, since at least their orders of magnitude can still be compared. We also give the number of subspace clusters found, the average dimensionality of these clusters, and their coverage (i.e., the percentage of objects of the dataset that were associated to clusters).

In order to illustrate the performances of Chameleoclust we focus on dataset *shape* in Table 2. For the sake of completeness the other results are given in Figure 3 . In Table 2, when an algorithm has a best possible run with a higher evaluation than Chameleoclust the result is highlighted in grey, and if the evaluation is similar to Chameleoclust then the result is simply emphasized in bold. The best possible runs of DOC and MINECLUS are observed with better results than standard runs of Chameleoclust for almost every quality measures, but they tend to split the dataset in more clusters (same behaviour also on the synthetic datasets) and have runtimes significantly higher than Chameleoclust. For PROCLUS and STAPC the best possible runs achieve better results than Chameleoclust for F1 and CE, but in these cases their coverage falls to about 80% to 90% leaving an important part of the dataset outside of the clusters. Looking at entropy many algorithms have best possible runs leading to a better *entropy* than Chameleoclust. However, in clustering tasks, the entropy cannot be interpreted regardless of the number of clusters, because usually the entropy quality measure tends to improve when the number of clusters increases. Indeed, by definition of the entropy measure, the best entropy is obtained for the extreme case where we have one cluster per object. Chameleoclust and three other algorithms (FIRES, P3C, STATPC) provide reasonable number of clusters with average quality entropy. As it can be expected, algorithms that increase importantly the number of clusters achieve higher entropy quality. Notice that Chameleoclust is the only one to obtain a reasonable number of clusters with a 100% coverage. Finally, the last row of the table shows the evaluations obtained when running Chameleoclust using only functional elements in the genome (i.e., g is constant and set to 1 in all genome elements $\langle g, c, d, x \rangle$). In this case we can observe a decrease of the performances that seems to advocate for the interest of the presence of both functional and non-functional elements, and that needs to be investigated further.

Figure 3 shows that for almost every dataset, the performances of Chameleoclust are reasonable with respect to the best possible runs of the other algorithms. Note that most of the time the number of classes within a dataset does not correspond to the number of clusters found by the algorithms. Indeed, predefined classes do not always form clusters. Consequently it is not surprising to obtain more clusters than classes. In some cases a few algorithms found a very large number of clusters (sometimes even more clusters than objects) and this is due to their ability to search for overlapping clusters. For all datasets, the number of clusters found by Chameleoclust is coherent with the number of clusters found by the other algorithms. Indeed Chameleoclust is able to adapt the number of clusters it produces and also their average dimensionality according to each dataset without specific parameter tuning, as summarized in Table 1.

Table 1: Average number of clusters and average dimensionality per cluster found for each dataset

$Dataset$	$NumClusters$	$AvgDim$
breast	15.2	5.32
diabetes	60.8	2.06
glass	34.8	2.67
liver	62.8	1.93
pendigits	17.5	5.61
shape	14.9	7.27
vowel	48.7	2.59

Synthetic data.

Chameleoclust was executed 10 times on each of the 16 synthetic datasets. For each dataset we kept the run reaching the highest fitness (for the best individual) among the 10 runs (notice that this selection is made without using any external labeling, but only the fitness values). Then for each evaluation measure, we plotted the measure value obtained with respect to the number of clusters found by each of the 16 selected runs. The results are shown Figures 4 to 8. For almost every synthetic dataset the number of clusters found by Chameleoclust is very close to the real number of clusters in the dataset (i.e., 10). Chameleoclust always found between 9 and 16 clusters, and for most of the datasets it found the exact number of clusters. As reported in [Müller et al., 2009] the other algorithms found between 5 and 50 clusters, excepted a few cases where much more clusters were found (up to more than several thousands). Among the other

Breast: 33 dimensions, 2 classes, 198 objects

	$F1$		$Accuracy$		CE		$\overline{RNIA}$		$\overline{Entropy}$		$Coverage$		$NumClusters$		$AvgDim$		$Runtime$	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0.67	0.67	0.71	0.71	0.02	0.02	0.40	0.40	0.26	0.26	1.00	1.00	107	107	1.7	1.7	453	453
DOC	0.73	0.61	0.81	0.76	0.11	0.04	0.84	0.07	0.46	0.27	1.00	0.80	60	6	27.2	2.8	1E+06	37515
MINECLUS	0.78	0.69	0.78	0.76	0.19	0.18	1.00	1.00	0.56	0.37	1.00	1.00	64	32	33.0	33.0	40359	29437
SCHISM	0.67	0.67	0.75	0.69	0.01	0.01	0.36	0.34	0.35	0.34	1.00	0.99	248	197	2.3	2.2	158749	114609
SUBCLU	0.68	0.51	0.77	0.67	0.02	0.01	0.54	0.04	0.27	0.24	1.00	0.82	357	5	2.0	1.0	5265	16
FIRES	0.49	0.03	0.76	0.76	0.03	0.00	0.05	0.00	1.00	0.01	0.76	0.04	11	1	2.5	1.0	250	31
INSCY	0.74	0.55	0.77	0.76	0.02	0.00	0.24	0.11	0.60	0.39	0.97	0.74	2038	167	11.0	4.4	134373	63484
PROCLUS	0.57	0.52	0.80	0.74	0.51	0.11	0.65	0.43	0.32	0.23	0.89	0.69	9	2	24.0	18.0	703	141
P3C	0.63	0.63	0.77	0.77	0.04	0.04	0.19	0.19	0.36	0.36	0.85	0.85	28	28	6.9	6.9	6281	6281
STATPC	0.41	0.41	0.78	0.78	0.16	0.16	0.33	0.33	0.29	0.29	0.43	0.43	5	5	33.0	33.0	5187	4906
Chameleoclust	0.64	0.58	0.8	0.76	0.15	0.09	0.32	0.23	0.31	0.25	1	1	18	12	12.08	4.61	566	523
mean	0.61		0.78		0.11		0.26		0.28		1		15.2		7.45		544	

Diabetes: 8 dimensions, 2 classes, 768 objects

	$F1$		$Accuracy$		CE		$\overline{RNIA}$		$\overline{Entropy}$		$Coverage$		$NumClusters$		$AvgDim$		$Runtime$	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0.70	0.39	0.72	0.69	0.03	0.01	0.14	0.01	0.23	0.13	1.00	1.00	349	202	4.2	2.4	11953	203
DOC	0.71	0.71	0.72	0.69	0.31	0.26	0.92	0.79	0.31	0.24	1.00	0.93	64	17	8.0	5.1	1E+06	51640
MINECLUS	0.72	0.66	0.71	0.69	0.63	0.13	0.89	0.58	0.29	0.17	0.99	0.96	39	3	6.0	5.2	3578	62
SCHISM	0.70	0.62	0.73	0.68	0.08	0.01	0.36	0.09	0.34	0.20	1.00	0.79	270	21	4.2	3.9	35468	250
SUBCLU	0.74	0.45	0.71	0.68	0.01	0.01	0.01	0.01	0.14	0.11	1.00	1.00	1601	325	4.7	4.0	190122	58718
FIRES	0.52	0.03	0.65	0.64	0.12	0.00	0.27	0.00	0.68	0.00	0.81	0.03	17	1	2.5	1.0	4234	360
INSCY	0.65	0.39	0.70	0.65	0.37	0.11	0.45	0.42	0.44	0.15	0.83	0.73	132	3	6.7	5.7	112093	33531
PROCLUS	0.67	0.61	0.72	0.71	0.34	0.21	0.78	0.69	0.23	0.19	0.92	0.78	9	3	8.0	6.0	360	109
P3C	0.39	0.39	0.66	0.65	0.56	0.11	0.85	0.22	0.09	0.07	0.97	0.88	2	1	7.0	2.0	656	141
STATPC	0.73	0.59	0.70	0.65	0.06	0.00	0.63	0.17	0.72	0.28	0.97	0.75	363	27	8.0	8.0	27749	4657
Chameleoclust	0.74	0.7	0.79	0.76	0.1	0.04	0.41	0.36	0.34	0.3	1	1	65	52	2.6	1.97	1956	1811
mean	0.72		0.77		0.07		0.38		0.32		1		60.8		2.24		1871	

Glass: 9 dimensions, 6 classes, 214 objects

	$F1$		$Accuracy$		CE		$\overline{RNIA}$		$\overline{Entropy}$		$Coverage$		$NumClusters$		$AvgDim$		$Runtime$	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0.51	0.31	0.67	0.50	0.02	0.00	0.06	0.00	0.39	0.24	1.00	1.00	6169	175	5.4	3.1	411195	1375
DOC	0.74	0.50	0.63	0.50	0.23	0.13	0.93	0.33	0.72	0.50	0.93	0.91	64	11	9.0	3.3	23172	78
MINECLUS	0.76	0.40	0.52	0.50	0.24	0.19	0.78	0.45	0.72	0.46	1.00	0.87	64	6	7.0	4.3	907	15
SCHISM	0.46	0.39	0.63	0.47	0.11	0.04	0.33	0.20	0.44	0.38	1.00	0.79	158	30	3.9	2.1	313	31
SUBCLU	0.50	0.45	0.65	0.46	0.00	0.00	0.01	0.01	0.42	0.39	1.00	1.00	1648	831	4.9	4.3	14410	4250
FIRES	0.30	0.30	0.49	0.49	0.21	0.21	0.45	0.45	0.40	0.40	0.86	0.86	7	7	2.7	2.7	78	78
INSCY	0.57	0.41	0.65	0.47	0.23	0.09	0.54	0.26	0.67	0.47	0.86	0.79	72	30	5.9	2.7	4703	578
PROCLUS	0.60	0.56	0.60	0.57	0.13	0.05	0.51	0.17	0.76	0.68	0.79	0.57	29	26	8.0	2.0	375	250
P3C	0.28	0.23	0.47	0.39	0.14	0.13	0.30	0.27	0.43	0.38	0.89	0.81	3	2	3.0	3.0	32	31
STATPC	0.75	0.40	0.49	0.36	0.19	0.05	0.67	0.37	0.88	0.36	0.93	0.80	106	27	9.0	9.0	1265	390
Chameleoclust	0.7	0.59	0.71	0.67	0.24	0.11	0.52	0.33	0.64	0.6	1	1	38	27	5.03	3.42	603	432
mean	0.62		0.68		0.19		0.46		0.62		1		34.8		4.17		544	

Liver: 6 dimensions, 2 classes, 345 objects

	$F1$		$Accuracy$		CE		$\overline{RNIA}$		$\overline{Entropy}$		$Coverage$		$NumClusters$		$AvgDim$		$Runtime$	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0.68	0.65	0.67	0.58	0.08	0.02	0.38	0.03	0.10	0.02	1.00	1.00	1922	19	4.1	1.7	38281	15
DOC	0.67	0.64	0.68	0.58	0.11	0.07	0.51	0.35	0.18	0.11	0.99	0.90	45	13	3.0	1.9	625324	1625
MINECLUS	0.73	0.63	0.65	0.58	0.09	0.09	0.68	0.48	0.33	0.16	0.99	0.92	64	32	4.0	3.7	49563	1954
SCHISM	0.69	0.69	0.68	0.59	0.04	0.03	0.45	0.26	0.10	0.08	0.99	0.99	90	68	2.7	2.1	31	0
SUBCLU	0.68	0.68	0.64	0.58	0.11	0.02	0.68	0.05	0.07	0.02	1.00	1.00	334	64	3.4	1.3	1422	47
FIRES	0.58	0.04	0.58	0.56	0.14	0.00	0.39	0.01	0.37	0.00	0.84	0.03	10	1	3.0	1.0	531	46
INSCY	0.66	0.66	0.62	0.61	0.03	0.03	0.42	0.39	0.21	0.20	0.85	0.81	166	130	2.1	2.1	407	234
PROCLUS	0.53	0.39	0.63	0.63	0.26	0.11	0.66	0.25	0.05	0.05	0.83	0.46	6	2	5.0	3.0	78	31
P3C	0.36	0.35	0.58	0.58	0.55	0.27	0.96	0.47	0.02	0.01	0.98	0.94	2	1	6.0	3.0	172	32
STATPC	0.69	0.57	0.65	0.58	0.23	0.01	0.58	0.37	0.63	0.05	0.77	0.71	159	4	6.0	3.3	1890	781
Chameleoclust	0.71	0.65	0.78	0.7	0.12	0.07	0.51	0.43	0.27	0.18	1	1	69	56	2.34	1.85	730	636
mean	0.68		0.75		0.10		0.46		0.21		1		62.8		2.04		703	

Pendigits: 16 dimensions, 10 classes, 7494 objects

	$F1$		$Accuracy$		CE		$\overline{RNIA}$		$\overline{Entropy}$		$Coverage$		$NumClusters$		$AvgDim$		$Runtime$	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0.30	0.17	0.96	0.86	0.06	0.01	0.20	0.06	0.41	0.26	1.00	1.00	1890	36	3.1	1.5	67891	219
DOC	0.52	0.52	0.54	0.54	0.18	0.18	0.35	0.35	0.53	0.53	0.91	0.91	15	15	5.5	5.5	178358	178358
MINECLUS	0.87	0.87	0.86	0.86	0.48	0.48	0.89	0.89	0.82	0.82	1.00	1.00	64	64	12.1	12.1	780167	692651
SCHISM	0.45	0.26	0.93	0.71	0.05	0.01	0.30	0.08	0.50	0.45	1.00	0.93	1092	290	10.1	3.4	5E+08	21266
SUBCLU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FIRES	0.45	0.45	0.73	0.73	0.09	0.09	0.33	0.33	0.31	0.31	0.94	0.94	27	27	2.5	2.5	169999	169999
INSCY	0.65	0.48	0.78	0.68	0.07	0.07	0.30	0.28	0.77	0.69	0.91	0.82	262	106	5.3	4.6	2E+06	1E+06
PROCLUS	0.78	0.73	0.74	0.73	0.31	0.27	0.64	0.45	0.90	0.71	0.90	0.74	37	17	14.0	8.0	6045	4250
P3C	0.74	0.74	0.72	0.72	0.28	0.28	0.58	0.58	0.76	0.76	0.90	0.90	31	31	9.0	9.0	2E+06	2E+06
STATPC	0.91	0.32	0.92	0.10	0.09	0.00	0.67	0.11	1.00	0.53	0.99	0.84	4109	56	16.0	16.0	5E+07	3E+06
Chameleoclust	0.7	0.51	0.73	0.59	0.4	0.25	0.6	0.47	0.67	0.56	1	1	19	14	7.44	4.64	12638	9760
mean	0.59		0.63		0.31		0.53		0.60		1		18		5.74		12284	

Vowel: 10 dimensions, 11 classes, 990 objects

	$F1$		$Accuracy$		CE		$\overline{RNIA}$		$\overline{Entropy}$		$Coverage$		$NumClusters$		$AvgDim$		$Runtime$	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0.23	0.17	0.64	0.37	0.05	0.00	0.44	0.01	0.10	0.09	1.00	1.00	3062	267	4.9	1.9	523233	1953
DOC	0.49	0.49	0.44	0.44	0.14	0.14	0.85	0.85	0.58	0.58	0.86	0.86	64	64	10.0	10.0	120015	120015
MINECLUS	0.48	0.43	0.37	0.37	0.09	0.04	0.62	0.34	0.60	0.46	0.98	0.87	64	64	7.2	3.6	7734	5204
SCHISM	0.37	0.23	0.62	0.52	0.05	0.01	0.43	0.11	0.29	0.21	1.00	0.93	494	121	4.3	2.8	23031	391
SUBCLU	0.24	0.18	0.58	0.38	0.04	0.01	0.39	0.04	0.30	0.13	1.00	1.00	10881	709	3.6	2.0	26047	2250
FIRES	0.16	0.14	0.13	0.11	0.02	0.02	0.14	0.13	0.16	0.13	0.50	0.45	32	24	2.1	1.9	563	250
INSCY	0.82	0.33	0.61	0.15	0.09	0.07	0.75	0.26	0.94	0.21	0.90	0.81	163	74	9.5	4.3	75706	39390
PROCLUS	0.49	0.49	0.44	0.44	0.11	0.11	0.53	0.53	0.65	0.65	0.67	0.67	64	64	8.0	8.0	766	766
P3C	0.08	0.05	0.17	0.16	0.12	0.08	0.69	0.43	0.13	0.12	0.98	0.95	3	2	7.0	4.7	1610	625
STATPC	0.22	0.22	0.56	0.56	0.06	0.06	0.12	0.12	0.14	0.14	1.00	1.00	39	39	10.0	10.0	18485	16671
Chameleoclust	0.45	0.37	0.49	0.42	0.11	0.08	0.42	0.36	0.47	0.43	1	1	56	31	3.02	2.66	2925	1947
mean	0.43		0.48		0.10		0.39		0.46		1		48.7		2.88		2618	

Figure 3: Result tables for six other real datasets

Table 2: Results for the Shape real dataset: 17 dimensions, 9 classes, 160 objects

	$F1$		$Accuracy$		CE		$\overline{RNIA}$		$\overline{Entropy}$		$Coverage$		$NumClusters$		$AvgDim$		$Runtime$	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0.31	0.31	**0.76**	**0.76**	0.01	0.01	0.07	0.07	0.66	0.66	**1.00**	**1.00**	486	486	3.3	3.3	235	235
DOC	0.90	0.83	**0.79**	**0.54**	**0.56**	**0.38**	0.90	0.82	0.93	0.86	**1.00**	**1.00**	53	29	13.8	12.8	2E+06	86500
MINECLUS	0.94	0.86	**0.79**	**0.60**	0.58	0.46	1.00	1.00	0.93	0.82	**1.00**	**1.00**	64	32	17.0	17.0	46703	3266
SCHISM	0.51	0.30	**0.74**	**0.49**	0.10	0.00	0.26	0.01	**0.85**	**0.55**	**1.00**	**0.92**	8835	90	6.0	3.9	712964	9031
SUBCLU	0.36	0.29	0.70	0.64	0.00	0.00	0.05	0.04	0.89	0.88	**1.00**	**1.00**	3468	3337	4.5	4.1	4063	1891
FIRES	0.36	0.36	0.51	0.44	0.20	0.13	0.25	0.20	0.88	0.82	0.45	0.39	**10**	**5**	7.6	5.3	63	47
INSCY	**0.84**	**0.59**	**0.76**	**0.48**	0.18	0.16	0.37	0.24	0.94	0.87	0.88	0.82	185	48	9.8	9.5	22578	11531
PROCLUS	0.84	0.81	0.72	0.71	0.25	0.18	**0.61**	**0.37**	0.93	0.91	0.89	0.79	34	34	13.0	7.0	**593**	**469**
P3C	0.51	0.51	0.61	0.61	0.14	0.14	0.17	0.17	0.80	0.80	0.66	0.66	**9**	**9**	4.1	4.1	140	140
STATPC	0.43	0.43	**0.74**	**0.74**	0.45	0.45	**0.55**	**0.55**	0.56	0.56	0.92	0.92	**9**	**9**	17.0	17.0	**250**	**171**
Chameleoclust	0.74	0.67	0.82	0.73	0.42	0.34	0.6	0.48	0.79	0.73	1	1	16	13	10	7.29	314	287
mean	0.71		0.79		0.38		0.54		0.76		1		15		8.26		309	
Only functional	0.73	0.53	0.8	0.66	0.45	0.18	0.65	0.28	0.77	0.62	1	1	15	9	11.33	3.78	366	159
mean	0.64		0.75		0.35		0.51		0.72		1		12		6.84		269	

algorithms, P3C is the one that gave the most often a number of clusters close to the real number, outputting between 6 and 16 clusters on these 16 synthetic datasets. Most of the evaluation measures for Chameleoclust are comparable to the ones reported in [Müller et al., 2009], keeping in mind that for the other algorithms only the best values of the evaluation measures over the parameter spaces were retained.

In addition, we give Figure 9 and Figure 10 the runtime of Chameleoclust with respect to the number of objects and number of dimensions of the synthetic datasets. The corresponding curves show that the algorithm scales rather linearly in both cases.

5. RELATED WORK

Many approaches have been investigated for subspace clustering in the literature using various clustering paradigms. The reader is referred for instance to [Kriegel et al., 2009], [Müller et al., 2009], and [Parsons et al., 2004] for detailed reviews and comparisons of the best methods and main categories:

- The cell-based approach, that defines clusters as hyper-rectangles laying in specific subspaces and containing more than a given number of objects. Clusters are usually constructed by discretizing the data space into axis-parallel cells and then aggregating promising cells. These selected cells are commonly the ones containing more objects than a threshold given as parameter. Other typical parameters are the number or the size of the cells.
- The density based approach, in which clusters are dense groups of objects in space. A cluster can have an arbitrary shape, but must be separated from the other clusters by low density regions. This approach defines dense regions as regions where within a given radius a number of objects exceeding a minimum threshold can be found. Clusters are built by joining together the objects from adjacent dense regions.
- The clustering-oriented approach, that usually defines properties of the targeted clustering such as the expected number of clusters or the cluster average dimensionality. According to these constraints, the objects are grouped together mainly using distance-based similarity. Most of these methods tend to build hyper-spherical shaped clusters in particular subspaces.

Even if many evolutionary clustering approaches exist [Hruschka et al., 2009] very few of them address the subspace clustering problem. An early approach was presented in [Sarafis et al., 2003], introducing a subspace clustering evolutionary algorithm that uses a rule-based representation to encode axis-parallel hyper-rectangular disjoint clusters. This algorithm is a member of the cell-based subspace clustering family. It uses task-specific mutation and recombination operators, and requires a non-evolutionary first stage to find promising clusters in 2D subspaces. More recently, in [Vahdat et al., 2010], a different evolutionary approach has been presented. It is also based on a first non-evolutionary clustering stage, used here to find a set

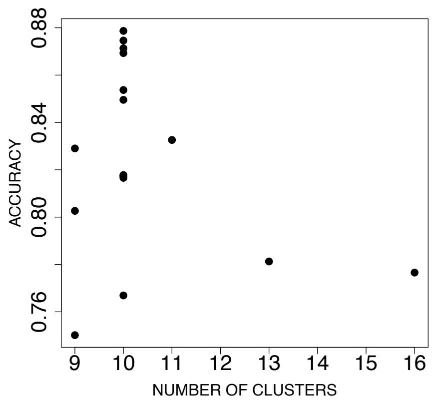

Figure 4: *Accuracy* and number of clusters for the best fitness among 10 runs for the 16 synthetic datasets.

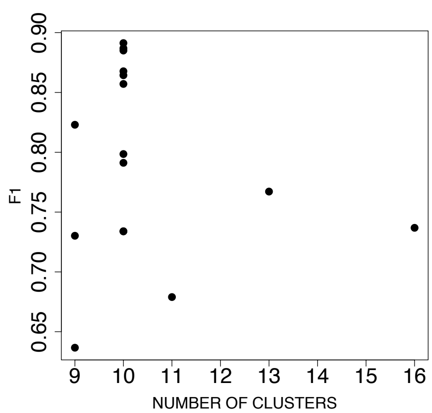

Figure 5: $F1$ and number of clusters for the best fitness among 10 runs for the 16 synthetic datasets.

of cluster candidate positions in each dimension. Next, it uses a genetic algorithm to produce subspace clusters by combining the candidate positions found at the previous step. The final stage is then to run a second genetic algorithm to find the best combination of subspace clusters to form the whole clustering of the data. This approach is related to the clustering-oriented family. The Chameleoclust algorithm presented in this paper also falls into the clustering-oriented category, but it is a single stage fully evolutionary approach, without any preliminary stage to identified clusters in lower dimensional spaces. In addition it relies on generic bio-like mutation operations that are not specific to the subspace clustering task. Moreover, Chameleoclust has been compared to state-of-the-art subspace clustering algorithms using a reference framework and performed well, while requiring a single and easy to tune parameter (the maximal number of desired clusters).

6. CONCLUSION

In this paper, we presented Chameleoclust, an evolutionary algorithm for subspace clustering. Its key underlying principle is to use an evolvable genome structure to find various numbers of clusters in subspaces of various dimensionality. It was shown to be competitive with respect to state-of-the-art algorithms using an evaluation framework of reference. Directions for future work

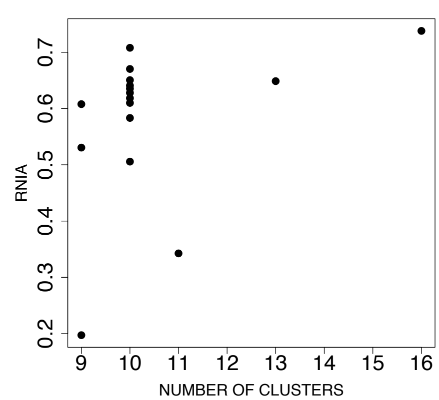

Figure 6: $\overline{RNIA}$ and number of clusters for the best fitness among 10 runs for the 16 synthetic datasets.

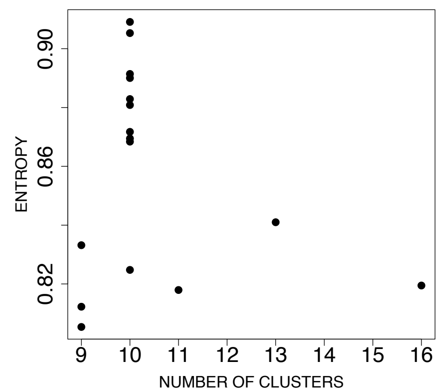

Figure 7: $\overline{Entropy}$ and number of clusters for the best fitness among 10 runs for the 16 synthetic datasets.

include: a deeper analysis of the benefits of the presence of the non-functional elements, the study of the stability of the parameters related to the evolution strategy (population size, mutation rate, ...), and the extension of the approach to try to take advantage of some crossover operations.

Acknowledgements

This research has been supported by EU-FET grant EvoEvo (ICT-610427).

7. REFERENCES

[Aggarwal et al., 2001] Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Proc. of the 8th Int. Conf. on Database Theory*, pages 420–434. Springer.

[Aggarwal et al., 1999] Aggarwal, C. C., Wolf, J. L., Yu, P. S., Procopiuc, C., and Park, J. S. (1999). Fast algorithms for projected clustering. In *Proc. of the 1999 ACM SIGMOD Int. Conf. on Management of Data*, pages 61–72, New York, NY, USA.

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). UCI machine learning repository.

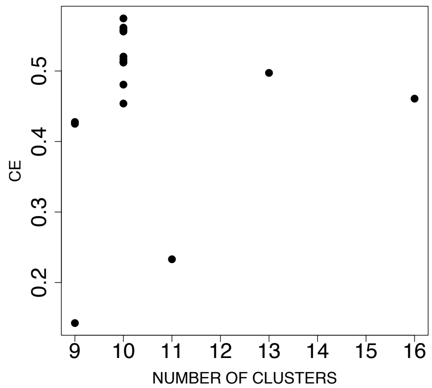

Figure 8: CE and number of clusters for the best fitness among 10 runs for the 16 synthetic datasets.

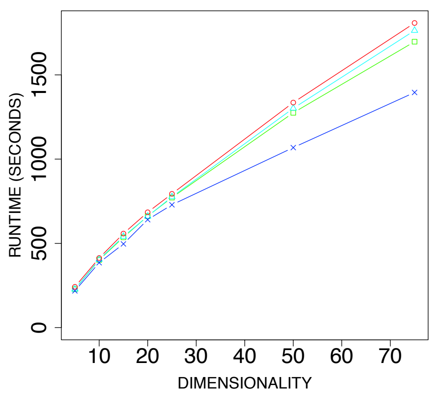

Figure 9: Runtime vs. dimensionality of the synthetic datasets. Maximal in red (circles), median in cyan (triangles), average in green (squares) and minimal in blue (crosses).

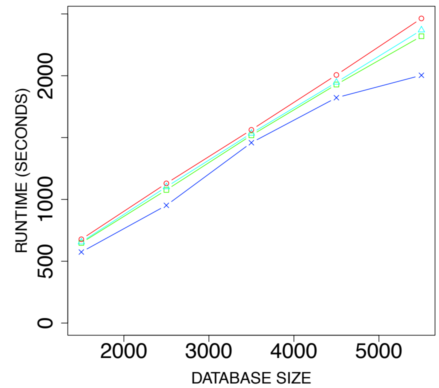

Figure 10: Runtime vs. number of objects of the synthetic datasets. Maximal in red (circles), median in cyan (triangles), average in green (squares) and minimal in blue (crosses).

[Banzhaf et al., 2006] Banzhaf, W., Beslon, G., Christensen, S., James, A., Képès, F., Lefort, V., Julian, F., Radman, M., and Ramsden, J. J. (2006). Guidelines: From artificial evolution to computational evolution: a research agenda. *Nature Reviews Genetics*, 7(9):729–735.

[Beyer et al., 1999] Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In *Proc. of the 7th Int. Conf. on Database Theory*, pages 217–235, London, UK.

[Crombach and Hogeweg, 2007] Crombach, A. and Hogeweg, P. (2007). Chromosome rearrangements and the evolution of genome structuring and adaptability. *Molecular Biology and Evolution*, 24(5):1130–9.

[Hindré et al., 2012] Hindré, T., Knibbe, C., Beslon, G., and Schneider, D. (2012). New insights into bacterial adaptation through in vivo and in silico experimental evolution. *Nature Reviews Microbiology*.

[Hruschka et al., 2009] Hruschka, E. R., Campello, R. J. G. B., Freitas, A. A., and de Carvalho, A. C. P. L. F. (2009). A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics*, 39(2):133–155.

[Knibbe et al., 2007] Knibbe, C., Coulon, A., Mazet, O., Fayard, J.-M., and Beslon, G. (2007). A Long-Term Evolutionary Pressure on the Amount of Noncoding DNA. *Molecular Biology and Evolution*, 24(10):2344–2353.

[Kriegel et al., 2009] Kriegel, H.-P., Kröger, P., and Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1:1–1:58.

[Müller et al., 2009] Müller, E., Günnemann, S., Assent, I., and Seidl, T. (2009). Evaluating clustering in subspace projections of high dimensional data. In *Proc. 35th Int. Conf. on Very Large Data Bases (VLDB 2009)*, volume 2, pages 1270–1281, Lyon, France.

[Parsons et al., 2004] Parsons, L., Haque, E., and Liu, H. (2004). Subspace clustering for high dimensional data: A review. *SIGKDD Explorations Newsletter*, 6(1):90–105.

[Patrikainen and Meila, 2006] Patrikainen, A. and Meila, M. (2006). Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering*, pages 902–916.

[Sarafis et al., 2003] Sarafis, I. A., Trinder, P. W., and Zalzala, A. (2003). Towards effective subspace clustering with an evolutionary algorithm. In *Proc. of the 2003 Congress on Evolutionary Computation (CEC-2003)*, volume 2, pages 797–806.

[Vahdat et al., 2010] Vahdat, A., Heywood, M. I., and Zincir-Heywood, A. N. (2010). Bottom-up evolutionary subspace clustering. In *IEEE Congress on Evolutionary Computation*, pages 1–8.