Subspace Clustering for all Seasons

Sergio Peignier[1], Christophe Rigotti[1], and Guillaume Beslon[1]

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

Subspace clustering is recognized as a more general and difficult task than standard clustering since it requires to identify not only objects sharing similar feature values but also the various subspaces where these similarities appear. Many approaches have been investigated for subspace clustering in the literature using various clustering paradigms. The reader is referred for instance to [5] for detailed reviews and comparisons of the best methods and main categories. Even if many evolutionary clustering approaches exist [4] very few of them address the subspace clustering problem and still include non-evolutionary stages [7,8].

According to [1], 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 operators (e.g., mutations) have on individuals, a phenomenon known as *evolution of evolution* [3]. 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 [2], and defined as a list of tuples (the "genes"), each tuple containing numbers. These tuples are mapped at the phenotype level to denote core point locations in different dimensions, which are then used to collectively build the subspace clusters, by grouping the data around the core points. The biological analogy here would be that each gene codes for a molecular product and that the combination of molecular products associated together codes for a function, i.e., a cluster. To allow for evolution of evolution, Chameleoclust genome contains a variable proportion of functional elements as , and is subject to local mutations and to large random rearrangements, namely: large deletions, duplications and translocations. Local mutations and rearrangements may thus modify the genome elements but also the genome structure. 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 and to self-tune the main evolutionary parameters (e.g., levels of variability). Figure 1 illustrates Chameleoclust genome structure (genome length and proportion of functional elements) and fitness convergence on a synthetic benchmark dataset. The reader is refered to [6] for a detailed description of Chameleoclust. The algorithm has been assessed using a reference framework for subspace clustering evaluation, and compared to state-of-the-art algorithms on both real and synthetic datasets [5]. However in [5] the parameters of the different state-of-the-art algorithms have been optimized according to external

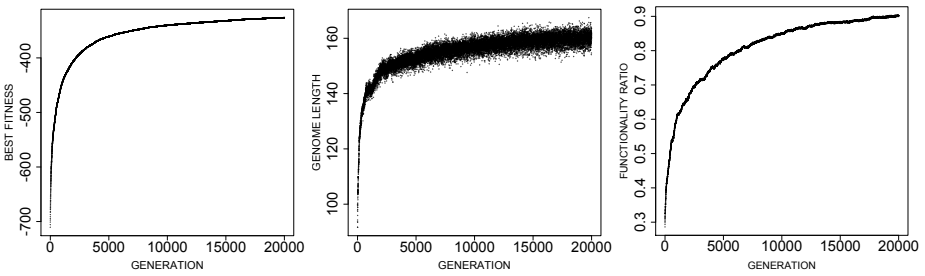

Figure 1: Average of the best fitness, genome length and proportion of functional elements (functionality ratio) of all individuals for 10 runs on a synthetic dataset (1500 objects, 10 dimensions, 0% noise).

information in order to compare the different algorithms by the best possible subspace clusterings they could achieve 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. This is one of the benefits evolution of evolution may offer. Consequently we decided not to perform any parameter optimization using external information for Chameleoclust. So, we compare clusterings effectively found by Chameleoclust to the best clusterings that could potentially be found by the other algorithms. For almost

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

	Accuracy		*CE*		*NumClusters*		*AvgDim*	
	max	min	max	min	max	min	max	min
CLIQUE	**0.76**	**0.76**	0.01	0.01	486	486	3.3	3.3
DOC	**0.79**	**0.54**	**0.56**	**0.38**	53	29	13.8	12.8
MINECLUS	**0.79**	**0.60**	0.58	0.46	64	32	17.0	17.0
SCHISM	**0.74**	**0.49**	0.10	0.00	8835	90	6.0	3.9
SUBCLU	0.70	0.64	0.00	0.00	3468	3337	4.5	4.1
FIRES	0.51	0.44	0.20	0.13	**10**	**5**	7.6	5.3
INSCY	**0.76**	**0.48**	0.18	0.16	185	48	9.8	9.5
PROCLUS	0.72	0.71	0.25	0.18	34	34	13.0	7.0
P3C	0.61	0.61	0.14	0.14	**9**	**9**	4.1	4.1
STATPC	**0.74**	**0.74**	0.45	0.45	**9**	**9**	17.0	17.0
Chameleoclust	0.82	0.73	0.42	0.34	16	13	10	7.29

Figure 2: Average number of clusters and average dimensionality per cluster found for each real world dataset (left). Results for the Shape real dataset: 17 dimensions, 9 classes, 160 objects (right)

every dataset, the performances of Chameleoclust are reasonable with respect to the best possible runs of the other algorithms (results on the real world dataset "Shape" are presented as an example in Figure 2). For all datasets, the number of clusters and the subspaces dimensionalities found by Chameleoclust are coherent with the number of clusters found by the other algorithms (e.g., Figure 2). 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 Figure 2. In addition, we give in Figure 3 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.

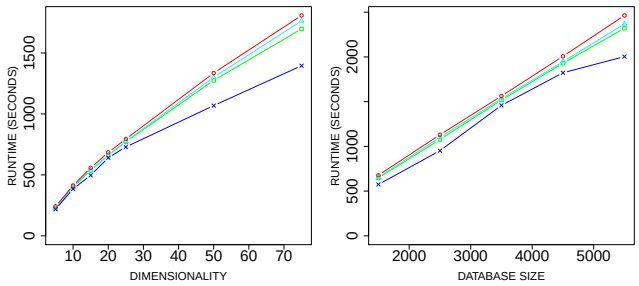

Figure 3: Runtime. Maximal in red (circles), median in cyan (triangles), average in green (squares) and minimal in blue (crosses).

The results obtained with the Chameleoclust algorithm show that evolution of evolution, through an evolvable genome structure, is usefull to solve a difficult problem such as subspace clustering.

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

References

1. Banzhaf, W., Beslon, G., Christensen, S., James, A., Képès, F., Lefort, V., Julian, F., Radman, M., Ramsden, J.J.: Guidelines: From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics 7(9), 729–735 (2006)
2. Crombach, A., Hogeweg, P.: Chromosome rearrangements and the evolution of genome structuring and adaptability. Molecular Biology and Evolution 24(5), 1130–9 (2007)
3. Hindré, T., Knibbe, C., Beslon, G., Schneider, D.: New insights into bacterial adaptation through in vivo and in silico experimental evolution. Nature Reviews Microbiology 10, 352–365 (May 2012)
4. Hruschka, E.R., Campello, R.J.G.B., Freitas, A.A., de Carvalho, A.C.P.L.F.: A survey of evolutionary algorithms for clustering. IEEE Transactions on Systems, Man, and Cybernetics 39(2), 133–155 (2009)
5. Müller, E., Günnemann, S., Assent, I., Seidl, T.: Evaluating clustering in subspace projections of high dimensional data. In: Proc. 35th Int. Conf. on Very Large Data Bases (VLDB 2009). pp. 1270–1281. Lyon, France (2009)
6. Peignier, S., Rigotti, C., Beslon, G.: Subspace clustering using evolvable genome structure. In: Proc. of the ACM Genetic and Evolutionary Computation Conference (GECCO 2015). pp. 1–8 (2015)
7. Sarafis, I.A., Trinder, P.W., Zalzala, A.: Towards effective subspace clustering with an evolutionary algorithm. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC 2003). pp. 797–806 (2003)
8. Vahdat, A., Heywood, M.I., Zincir-Heywood, A.N.: Bottom-up evolutionary subspace clustering. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC 2010). pp. 1–8 (2010)