IMPROVEMENTS FOR COPTBEES CLUSTERING ALGORITHM

Eugênio Monteiro da Silva Júnior* Renato Dourado Maia* Dávila Patrícia Ferreira Cruz† Leandro Nunes de Castro†

*Computer Science Department
State University of Montes Claros
Montes Claros, MG, Brazil

†Natural Computing Laboratory (LCoN)
Graduate Program in Electrical Engineering
Mackenzie University, São Paulo, SP, Brazil

Email: eugeniomonteiro@hotmail.com renato.dourado@unimontes.br
davilapatricia11@gmail.com lnunes@mackenzie.br

Abstract— Clustering is one of the most important tasks in data mining and can be defined as the process of segmenting heterogeneous data into a number of more homogeneous subgroups or clusters. There are several algorithms for this purpose, each of them with specific features. Among these algorithms is the cOptBees, a bee-inspired algorithm to solve data clustering problems. This algorithm have some advantages like the capacity of detecting the number of clusters automatically. However, the cOptBees has some drawbacks like high processing time for datasets with large number of samples. This paper proposes modifications for this algorithm in order to make it more efficient. The new version was applied to six datasets and the results indicate that the modifications improved the quality of the results and reduced the convergence time when compared to the previous version.

Keywords— Data Clustering; Swarm Intelligence; Bee-inspired Algorithms.

Resumo— O agrupamento é uma das mais importantes tarefas da mineração de dados e pode ser definida como o processo de segmentação de dados heterogêneos em um número de grupos mais homogêneos. Existem muitos algoritmos para esse propósito, cada um deles com características específicas. Entre esses algoritmos está o cOptBees, um algoritmo inspirado no comportamento das abelhas para resolver problemas de agrupamento. Esse algoritmo apresenta algumas vantagens como a capacidade de detectar automaticamente a quantidade de grupos. Entretanto, o cOptBees apresenta alguns inconvenientes como o elevado tempo de processamento para bases de dados com grande número de amostras. Esse artigo propõe modificações para esse algoritmo com o objetivo de torná-lo mais eficiente. A nova versão foi aplicada a seis bases de dados e os resultados indicaram que as modificações melhoraram a qualidade dos resultados e reduziram o tempo de convergência em relação à versão anterior.

Palavras-chave— Agrupamento, Inteligência de enxame, algoritmos inspirados em abelhas.

1 INTRODUCTION

Clustering is the task of segmenting a heterogeneous data into a number of more homogeneous subgroups or clusters. In clustering, there are no predefined classes. The records are grouped together on basis of self similarity (Berry and Linoff, 2004). When the objects are well allocated into a cluster, they are more similar to one another than to those belonging to other clusters (Han and Kamber, 2006). Clustering is often done as a preliminary step to some other forms of data mining or modeling. For example, clustering might be the first step in a market segmentation effort: first divide the customer base into clusters of people with similar buying habits, and then ask what kind of promotion works best for each cluster (Berry and Linoff, 2004).

This paper proposes a local search step for the cOptBees clustering algorithm proposed in (Cruz, Maia and de Castro, 2013). In addition to the local search, modifications were proposed in the initialization and in the similarity measure of the candidate solutions produced by the algorithm. These modifications aim to reduce the convergence time and improve the quality of the results.

This paper is organized as follows: Section 2 presents cOptBees originally developed in (Cruz, Maia and de Castro, 2013), an algorithm inspired by the bee foraging behavior to solve optimal data clustering problems; Section 3 details all changes proposed for the algorithm; Section 4 shows the experimental results of the comparison between the original and modified versions of the algorithm applied to several datasets; and Section 5 presents the conclusion and points out proposals for future works.

2 COPTBEES: A CLUSTERING ALGORITHM INSPIRED BY BEE COLONIES

The clustering algorithm named cOptBees was proposed in (Cruz, Maia, Szabo and de Castro, 2013). This algorithm is an adaption of Opt-

Bees, originally designed to solve continuous optimization problems (Maia et al., 2012).

The OptBees algorithm, summarized in Algorithm 1, was tested in the twenty-five minimization problems proposed by the 2005 IEEE Congress on Evolutionary Congress (CEC) (Suganthan et al., 2005). The results showed that OptBees is able to generate and maintain diversity of candidate solutions, finding multiple local optima without compromising its global search capability (Cruz, Maia and de Castro, 2013). According to (Cruz, Maia and de Castro, 2013), the main features and behavior of OptBees suggest it can be successfully adapted to solve other kinds of problems, such as clustering, where the generation and maintenance of diversity are important. The first version of cOptBees was presented in (Cruz, Maia, Szabo and de Castro, 2013) and shown to be able to generate and maintain diversity of solutions by finding multiple suboptimal solutions in a single run, a useful feature for solving multimodal optimization problems, such as optimal data clustering.

In cOptBees, the active bees can belong to one of three groups, according to their task: 1) recruiters, who attract other bees to explore a promising region of the search space; 2) recruited, who are recruited by recruiters to explore a promising region of the search space; or 3) scout bees, who randomly look for new promising regions of the space (Cruz, Maia, Szabo and de Castro, 2013).

Algorithm 1: OptBees

Input Parameters:

- n_{min}: minimum number of active bees.
- n_{max}: maximum number of active bees.
- ρ: inhibition radius.
- α: recruitment rate.
- ε: foraging effort.
- p_{min}: minimum probability of a bee being a recruiter.
- p_{rec}: percentage of non-recruiters that will be actually recruited.

Output Parameters:

- Active bees and their respective values of objective function.

```
begin
    Ramdomly generate a swarm of n bees;
    while stopping criterion is not attained
    do
        Evaluate the quality of the sites
        being explored by the active bees;
        Apply local search;
        Determine the recruiter bees;
        Update the number of active bees;
        Determine the recruited and scout
        bees;
        Perform the recruitment process;
        Perform the exploration process;
    end
end
```

3 IMPROVEMENTS

To identify the different versions of cOptBees, the following nomenclature was adopted: cOptBees_1, the version presented in (Cruz, Maia, Szabo and de Castro, 2013); cOptBees_2, the version presented in (Cruz, Maia and de Castro, 2013); cOptBees_3, the version presented in this work; and $\text{cOptBees}_{3\text{LS}}$, the cOptBees_3 with local search. This paper addresses only the comparison between cOptBees_2, cOptBees_3, and $\text{cOptBees}_{3\text{LS}}$ because these versions work with the same encoding scheme.

Four key modifications are introduced in cOptBees_3, as follows: 1) a new initialization scheme; 2) an exploration process based on appropriately choosing prototypes so as to find the groups medoids; 3) a dissimilarity measure that takes into account the distance among prototypes; and 4) a local search procedure that determines the medoid of each each cluster.

3.1 Initialization

Unlike the cOptBees_2, where the initial swarm is generated by random numbers of the search space, for this paper, each initial bee is composed of random members of the dataset (prototype), respecting the maximum number of clusters. Like cOptBees_2, for each prototype, there is an additional random value ($[0, 1]$) that represents the activation threshold. If the threshold L_j is greater than or equal to 0.5, the medoid C_j is active.

3.2 Exploration process

In the exploration process of cOptBees_2, the scout bees are moved to a random position in the search space. Like the swarm initialization proposed in this paper, in the proposed exploration process, the scout bees are moved to a new random set of prototypes and each activation threshold is also randomized.

3.3 Similarity measure

The new method to determine the similarity of solutions aims to reduce the processing time. The similarity measure utilized in cOptBees_2 has a

computational cost proportional to the dataset size. This process involves labeling the dataset according to each solution and then comparing the labels.

For this work, the similarity measure does not depend on the dataset, because only the active centroids of each solution are considered. Assume a comparison between a solution a and a solution b. Each of these solutions have a matrix of centroids that, according to the activation threshold, can be active or inactive. The method selects only the active prototypes (activation threshold greater than or equal to 0.5) of each solution. The algorithm calculates the Euclidean distance of each active prototype of a to each active prototype of b and computes the nearest ones. The similarity measure will be the average of the distance between the nearest prototypes of a and b. The solutions may not have the same number of prototypes, so it was added a penalty of 10% multiplied by the difference between the number of prototypes of a and b. Thus, if the solutions a and b have the same number of prototypes in the same positions, the value of similarity will be zero. Conceptually, this measure represents the dissimilarity between two solutions.

Assuming that the number of prototypes of each possible solution varies between 1 and the input parameter $rMax$, the time complexity of comparing two solutions is $O(rMax^2)$. The previous version (Cruz, Maia and de Castro, 2013) works by grouping a dataset of N elements; therefore, its time complexity is $O(N \times rMax)$. By definition, $rMax$ is always less than N (dataset size), therefore, the new method reduces the running time when compared to the previous one. This similarity measure was utilized when comparing two recruiters.

In the recruitment process, the recruiter bees recruit the non-recruiters with higher similarity to it. To determine this similarity, the matrix norm of the difference between the recruiter matrix and the non-recruiter matrix was utilized.

3.4 Local Search

One of the main objectives of this work is to propose a low cost local search for cOptBees$_2$, since this step was considered in (Cruz, Maia, Szabo and de Castro, 2013), but not used in (Cruz, Maia and de Castro, 2013). For this purpose, a simple algorithm was implemented. This algorithm works for all recruiter bees at each iteration. Initially, the dataset is labeled according to all prototypes of a solution, ignoring the activation threshold. Thereafter, there are three possibilities: do nothing; inactivate the active prototypes that do not group any element and inactivate some random active prototypes; or activate some random inactive prototypes that group some elements. These possibilities are randomized from an uniform distribution. After this step, the dataset is grouped according to the new set of active prototypes and, for each cluster, the algorithm calculates the medoids. If the new set of medoids represents improvements to the fitness (modified Silhouette), the matrix of centroids of the current solution is replaced by the new matrix; otherwise, the current solution is kept unchanged.

4 EXPERIMENTAL RESULTS

The experiments performed to evaluate the new version of cOptBees were similar to those presented in (Cruz, Maia, Szabo and de Castro, 2013) and (Cruz, Maia and de Castro, 2013). The algorithm was implemented in Matlab® and applied to the same five datasets used in the previous works: Iris, Wine, Balance, Haberman, and Ecoli (Lichman, 2013). The quality of solutions was evaluated by two measures: Entropy (E) and Purity (P). Entropy measures the homogeneity of a clustering solution, showing how the elements are distributed over the groups (Szabo et al., 2012): the lower the entropy, the better the quality of the clustering. Purity, by contrast, indicates how pure is a group, i.e., the ratio of the dominant class of a group in relation to the total number of elements in the group: the higher the purity, the better the quality (Szabo et al., 2012). In addition to these measures, the running time (T) in seconds was also evaluated.

All tests were performed in a computer with CPU Intel® Core i3, 4GB of RAM, and operating system Windows 7 64 bits.

All versions of cOptBees use the modified Silhouette (Russeeuw, 1987) as fitness function, which is calculated by:

$$s(x_i) = \frac{b(x_i) - a(x_i)}{max\{a(x_i), b(x_i)\}}, \qquad (1)$$

where $a(x_i)$ represents the dissimilarity between x_i and its centroid, and $b(x_i)$ represents the smaller dissimilarity between x_i and the centroid of other cluster (Hruschka et al., 2004). This function should be maximized.

4.1 Bidimensional Dataset

The previous works (Cruz, Maia, Szabo and de Castro, 2013) and (Cruz, Maia and de Castro, 2013) presented results for the cOptBees applied to the Ruspini data (Ruspini, 1970). However, this dataset is not very challenging and, therefore, the comparison for this dataset was suppressed.

For this work, a dataset known as R15 (Veenman et al., 2002) was chosen, available at the web page of the School of Computing of the University of Eastern Finland (Joensuu Clustering

Datasets). R15 is a synthetic dataset composed of 15 groups with 40 elements generated by a 2D Gaussian distribution, which totals 600 elements (Fig. 1).

The algorithms were run fifty times for the R15 dataset with the following input parameters: $n_{min} = 50$; $n_{max} = 100$; $rMax = 30$ (double of the exact number of clusters); $\varepsilon = 10$; $p_{min} = 0.01$; p_{rec} linearly varying between 0.1 and 0.5 with the number of iterations; ρ linearly varying between 0.1 and 0.3 with the number of iterations; number of iterations = 50; and $\alpha = 0.5$. The cOptBees$_3$ utilizes another concept of similarity measure, thus, the value was fixed in $\rho = 0.003$. The results of these tests can be observed in Table 1. This table shows the mean and standard deviation of each quality measure and the running time in seconds for the cOptBees$_2$, cOptBees$_3$, and cOptBees$_{3LS}$.

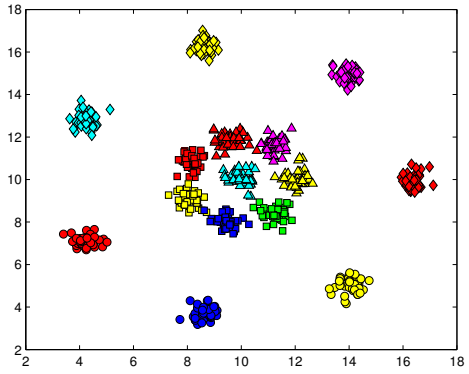

Figure 1: R15 dataset

Table 1: Mean ± Standard Deviation of results obtained by each algorithm when applied to the R15 dataset

	cOptBees$_2$	cOptBees$_3$	cOptBees$_{3LS}$
Fitness	0.70 ± 0.02	0.80 ± 0.03	0.81 ± 0.03
Entropy	0.10 ± 0.04	0.03 ± 0.02	0.02 ± 0.01
Purity	0.77 ± 0.11	0.93 ± 0.06	0.96 ± 0.04
N. clusters	12.44 ± 2.36	15.34 ± 1.92	16.06 ± 1.93
Time	1215.94 ± 56.9	596.2 ± 15.72	743.3 ± 12.0

According to the processing time, it is clear that the cOptBees$_3$ is faster than cOptBees$_2$. However, to assess the quality of the solutions obtained by the cOptBees$_3$, the Wilcoxon signed-rank test was performed in the results of entropy and purity with 5% of significance level. This test was performed for the three algorithms, two by two, and rejected the null hypothesis H_0 for all cases, which indicates that there are significant differences between the results. Fig. 2 and 3 shows the box plot for the entropy and purity values utilized by the test. These results allow us to conclude that the changes proposed in this work makes the algorithm not only faster, but also more accurate.

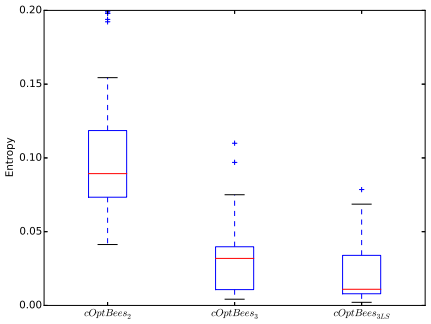

Figure 2: Box plot of the results of entropy for R15 dataset

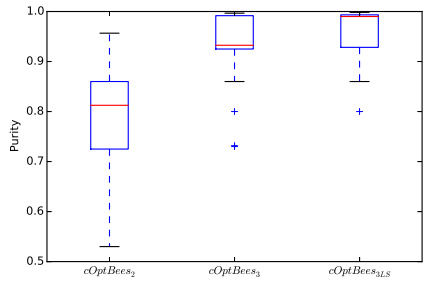

Figure 3: Box plot of the results of purity for R15 dataset

4.2 Multidimensional Datasets

As in (Cruz, Maia, Szabo and de Castro, 2013) and (Cruz, Maia and de Castro, 2013), the algorithm was tested in five datasets from the UCI Machine Learning Database (Lichman, 2013). Table 2 summarizes the main features of these datasets. As done in (Cruz, Maia and de Castro, 2013) and (Szabo et al., 2012), for the Ecoli dataset classes cp, im, imu, om, and pp were considered.

Table 2: Main features of the data sets used for performance comparison

Dataset	Number of elements	Number of attributes	Number of classes
Iris	150	4	3
Wine	178	13	3
Balance	625	4	3
Haberman	306	3	2
Ecoli	327	7	5

To evaluate the performance of the cOptBees, each version of the algorithm was run 10 times for each dataset with the following input paramaters: $n_{min} = 100$; $n_{max} = 200$; $\varepsilon = 10$; $p_{min} = 0.01$;

p_{rec} linearly varying between 0.1 and 0.5 with the number of iterations; number of iterations = 50; and $\alpha = 0.5$. For cOptBees$_2$, ρ linearly varying between 0.1 and 0.5 with the number of iterations. For cOptBees$_3$, ρ was fixed in 0.02, because the concept of similarity differ from one version to another. The number of executions and the parameters values are the same as those used in (Cruz, Maia and de Castro, 2013).

For cOptBees, it is necessary to inform the maximum number of clusters. As done in (Cruz, Maia and de Castro, 2013), this value was defined as twice the right number of clusters. So, for the Iris, Wine, and Balance datasets it was used $rMax = 6$, for Haberman, $rMax = 4$ and, for Ecoli, $rMax = 10$.

Table 3 presents the mean and the standard deviation of Entropy (E), Purity (P), number of clusters (C), and running time in seconds (S) for each dataset. The minimum running time was highlighted.

Table 3: Mean ± Standard Deviation of Entropy (E), Purity (P), number of clusters (C), and running time in seconds (T) for cOptBees$_2$, cOptBees$_3$, and cOptBees$_{3LS}$.

Dataset		cOptBees$_2$	cOptBees$_3$	cOptBees$_{3LS}$
Iris	E	0.10 ± 0.0	0.10 ± 0	0.10 ± 0
	P	0.66 ± 0	0.66 ± 0	0.66 ± 0
	C	2 ± 0	2 ± 0	2 ± 0
	T	614.7 ± 60.8	**82.19 ± 3.77**	147.0 ± 2.2
Wine	E	0.12 ± 0	0.13 ± 0	0.12 ± 0.02
	P	0.65 ± 0	0.62 ± 0	0.65 ± 0.09
	C	2 ± 0	2 ± 0	2.1 ± 0.31
	T	585.3 ± 42	**99.41 ± 4.47**	165 ± 5.1
Balance	E	0.14 ± 0	0.14 ± 0	0.14 ± 0
	P	0.66 ± 0.04	0.68 ± 0.03	0.67 ± 0.04
	C	3.8 ± 1.31	3.6 ± 0.84	4.2 ± 0.91
	T	1,470 ± 68.9	**314.31 ± 13.06**	502.7 ± 23.4
Haberman	E	0.10 ± 0	0.10 ± 0	0.10 ± 0
	P	0.73 ± 0	0.74 ± 0	0.74 ± 0
	C	2 ± 0	2.2 ± 0.42	2.2 ± 0.42
	T	144.4 ± 10.3	**90.60 ± 1.9**	99.5 ± 11.3
Ecoli	E	0.17 ± 0.01	0.12 ± 0	0.12 ± 0
	P	0.65 ± 0.04	0.77 ± 0	0.77 ± 0
	C	2.1 ± 0.3	3.1 ± 0.31	3 ± 0
	T	925.6 ± 42	**625.61 ± 51.93**	1,061 ± 38.6

The Wilcoxon sign-rank test with 5% of significance level rejected the null hypothesis only for Ecoli dataset, which indicates that the cOptBees$_3$ had better results when compared to cOptBees$_2$ for this dataset. For the others datasets, the quality of the results of cOptBees$_2$, cOptBees$_3$, and cOptBees$_{3LS}$ are statistically equivalent. However, the convergence time of the cOptBees$_3$ is smaller than that of cOptBees$_2$. In all cases, the cOptBees$_{3LS}$ reached fitness values higher than those of cOptBees$_2$. To investigate the convergence of cOptBees$_2$ and cOptBees$_{3LS}$, the mean and the standard deviation of the fitness at each iteration was plotted. As can be seen in Fig. 4, for the Haberman dataset, the cOptBees$_{3LS}$ exceeded the fitness value of cOptBees$_2$ before the 20th iteration. For the Balance dataset, cOptBees$_3$ reached higher fitness than that of cOptBees$_2$ before the 10th iteration (Fig. 5). For the Iris, Wine, and Ecoli, cOptBees$_{3LS}$ already started with a fitness better than cOptBees$_2$ and it is not overcome until the last iteration (Fig. 6, 7, and 8).

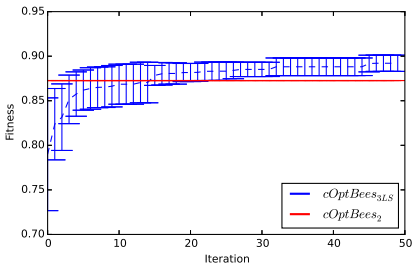

Figure 4: Fitness evolution for Haberman dataset

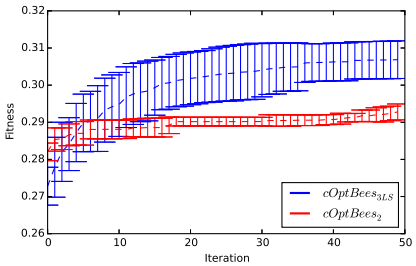

Figure 5: Fitness evolution for Balance dataset

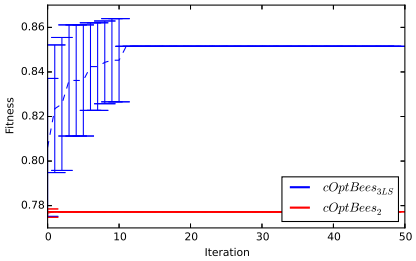

Figure 6: Fitness evolution for Iris dataset.

5 CONCLUSIONS

The results suggest that the proposed modifications improve the algorithm with respect to processing time, without a considerable loss of quality of the solutions and, in some cases, there was an improvement in quality as well. For the bidimensional dataset R15, these modifications allowed the algorithm to achieve better results with lower processing times. For the multimensional datasets, cOptBees$_3$ was statistically better than cOptBees$_2$ only for Ecoli. However, for the other four multidimensional datasets, the cOptBees$_3$ converged faster than cOptBees$_2$ and the average

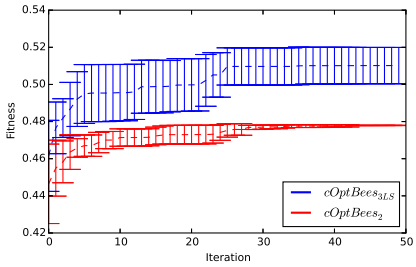

Figure 7: Fitness evolution for Wine dataset

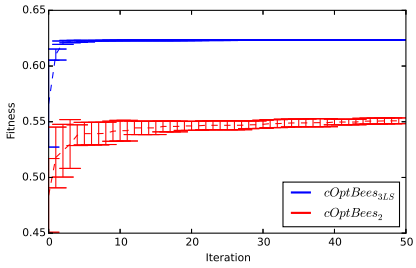

Figure 8: Fitness evolution for Ecoli dataset

processing time was much lower. These results indicate that there is a need for a more detailed study to determine the contexts that the modified version does not significantly affect the quality of solutions.

As the proposed modifications represented improvements for the cOptBees at the clustering task, as future research, we intend to utilize this algorithm as the unsupervised phase of RBF neural network training. There is a RBF classifier generated by cOptBees$_2$, named BeeRBF (Cruz et al., 2016). We intend to replace the cOptBees$_2$ by cOptBees$_3$ in the unsupervised training and perform the same tests performed in (Cruz et al., 2016) to verify if this new version produces also results significantly better in the classification task.

References

Berry, M. J. A. and Linoff, G. S. (2004). *Data Mining Techniques for Marketing, Sales and Customer Relationship Management*, 2nd edn, Wiley.

Cruz, D. P. F., Maia, R. D., da Silva, L. A. and de Castro, L. N. (2016). Beerbf: A bee-inspired data clustering approach to design rbf neural network classifier, *Neurocomputing* .

Cruz, D. P. F., Maia, R. D. and de Castro, L. N. (2013). A new encoding scheme for a bee-inspired optimal data clustering algorithm, *Proc. BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICSCCI and CBIC)*, Ipojuca.

Cruz, D. P. F., Maia, R. D., Szabo, A. and de Castro, L. N. (2013). A bee-inspired algorithm for optimal data clustering, *Proc. IEEE Congress on Evolutionary Computation*, Cancun.

Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*, Elsevier, San Francisco.

Hruschka, E. R., de Castro, L. N. and Campello, R. J. G. B. (2004). Evolutionary algorithms for clustering gene-expression data, *IEEE World Congress on Data Mining*, pp. 403–406.

Lichman, M. (2013). UCI machine learning repository.

Maia, R. D., de Castro, L. N. and Caminhas, W. M. (2012). Bee colonies as model for multimodal continuous optimization: The optbees algorithm, *IEEE World Congress on Computational Intelligence*.

Ruspini, H. R. (1970). Numerical methods for fuzzy clustering, *Information Sciences* .

Russeeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, *Journal of Computational and Applied Mathematics* **20**.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A. and Tiwari, S. (2005). Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization, *Technical report*, Singapore.

Szabo, A., de Castro, L. N. and Delgado, M. R. (2012). Fainet: an immune algorithm for fuzzy clustering, *IEEE International Conference on Fuzzy Systems*.

Veenman, C. J., Reinders, M. J. T. and Backer, E. (2002). A maximum variance cluster algorithm, *IEEE Trans. Pattern Analysis and Machine Intelligence* **24**: 1273–1280.