OCOPT-AINET: AN ARTIFICIAL IMMUNE SYSTEM FOR OPTIMAL CLUSTERING[1]

Ederson Borges*
Daniel Gomes Ferrari**
Leandro Nunes de Castro***

Abstract

Clustering is an important data mining task from the field of knowledge discovery in databases. Many algorithms can perform clustering in a simple and efficient manner, but have drawbacks, such as the lack of a way to automatically determine the optimal number of clusters in the dataset and the possibility of getting stuck in local optima solutions. To try and reduce these drawbacks this work proposes a new clustering algorithm based on artificial immune systems. This algorithm is characterized by the generation of multiple simultaneous high quality solutions in terms of the number of clusters in the database and the use of a cost function that explicitly evaluates the quality of clusters, minimizing the inconvenience of getting stuck in local optima solutions.

Keywords: Artificial immune systems, data clustering, ocopt-aiNet.

* Universidade Presbiteriana Mackenzie (UPM).
** Universidade Presbiteriana Mackenzie (UPM).
*** Universidade Presbiteriana Mackenzie (UPM).
[1] The authors thank Mackpesquisa, Fapesp and CNPq for the financial support.

1 INTRODUCTION

Natural computing has as one of its goals to develop solutions to real world problems by observing the natural principles and the behavior of species, including techniques that use only theories related to biology, for example, the theory of evolution. Researchers have used this theory to formulate the first *evolutionary algorithms* used in computational tasks, such as data clustering, pattern recognition and function optimization (YU; GEN, 2010). This work focuses on the development and application of one bio-inspired technique within the *artificial immune systems* field for the problem of grouping data.

Taken as an important task of knowledge discovery in databases, data clustering can be used to identify relationships of practical relevance between the data (JAIN; MURTY; FLYNN, 1999), such as the identification of customer profiles, anomaly detection and market segmentation, among others. A difficulty of many clustering techniques is the fact that they may need to be provided, a priori, a number *k* of possible clusters in the data base. The variation of several techniques, including *evolutionary algorithms*, has been introduced to approach a better solution for the automatic discovery of a suitable number of groups in a dataset (HRUSCHKA et al., 2009; SHEIKH; RAGHUWANSHI; JAISWAL, 2008; YANG, 1993; XU; WUNSCH, 2005).

It is therefore of great importance to develop new techniques to provide solutions that improve the efficiency of clustering techniques, especially regarding the automatic determination of the number of clusters. This paper aims to propose a new artificial immune algorithm for data clustering, which has the following main features:

- The automatic determination of the number *k* of clusters in a database;
- The use of an explicit cost function in the search process; and
- The maintenance of diversity through an explicit mechanism for identifying and pruning similar solutions.

This paper is organized as follows. Section 2 introduces the two algorithms that, when combined, generate the present proposal. In section 3 the new algorithm is proposed for the task of clustering. Section 4 shows the experiments and their results compared with other algorithms in the literature. And in Section 5 has the conclusion of the results and perspectives for future work.

2 THE PRECURSORS: OPT-AINET AND EAC

The algorithm to be proposed in this paper, named ocopt-aiNet, was designed by combining two other algorithms from the literature, an immune network named opt-aiNet and originally designed to solve multimodal optimization problems, and an evolutionary algorithm for clustering, named EAC. This section is devoted to the review of these two methods so that the reader can properly understand ocopt-aiNet.

2.1 Opt-aiNet: an immune network algorihtm for optmization

Many inspirations for the development of computational models and methods for solving complex problems can be found in biology. *Artificial immune systems* (AIS) take inspiration from the vertebrate immune system to develop computational tools for solving problems (HRUSCHKA et al., 2009; DASGUPTA; JI; GONZALEZ, 2003; CASTRO; TIMMIS, 2002).

A model based on *immune networks*, specifically *discrete networks* (CASTRO; TIMMIS, 2003), is the *aiNet* algorithm (CASTRO; VON ZUBEN, 2001). This algorithm was successfully applied to several problems in data compression and clustering. An optimization version of the *aiNet* algorithm, called *opt-aiNet* (CASTRO; TIMMIS, 2002), was developed with the ability to perform unimodal and multimodal searches (SHEIKH; RAGHUWANSHI; JAISWAL, 2008; CASTRO; TIMMIS, 2002).

In *opt-aiNet* each network cell represents a solution to the problem being treated. The algorithm is used to find solutions to continuous optimization problems, in which each cell is a real valued vector in an Euclidean space. The *opt-aiNet* algorithm uses an evaluation function to be optimized, which also provides the fitness value of that cell. The fitness measures the quality of the solution represented by a cell: high fitness values indicate good solutions, whilst low fitness values indicate lower quality solutions.

There are also the clone generation and mutation processes: at each generation there is only cell proliferation of a number of clones defined by the user, and in the mutation there is the variation of these clones, based on a specific criterion. In *opt-aiNet* the mutation rate is proportional to the fitness: cells with high fitness values suffer low mutation rates, whilst cells with low fitness suffer high mutation rates.

In addition to the fitness of a cell, which measures its quality in relation to a cost function to be optimized, a cell also has an affinity, representing how similar it is to other cells in the network. The affinity of a cell may lead to a cell cloning or pruning.

The affinity of cells is evaluated based on the Euclidean distance among them. Cells with an affinity greater than a pre-defined threshold may be pruned. After this suppression process new randomly generated cells are included in the network (SHEIKH; RAGHUWANSHI; JAISWAL, 2008; CASTRO; TIMMIS, 2002).

2.2 EAC: an evolutionary algorithm for clustering

The *evolutionary algorithms* have their inspirationin the Darwinian Theory of Evolution. These algorithms are able to find good solutions to complex problems in reasonable computational time. The *evolutionary algorithm for clustering* (EAC) was proposed to achieve optimal groupings of data (HRUSCHKA; CAMPELLO; CASTRO, 2006).

The *EAC* is started by a population of individuals, randomly generated, which represent candidate solutions to the data clustering problem. This initial generation is then used to produce offspring through preselected random variations. The resulting candidate solutions will be evaluated based on their effectiveness in solving the problem.

As the same way that the environment makes a selective pressure on individuals, in the evolutionary algorithm there is also a process where only the most adapted individuals (best *fitness*) are preserved to the next generation, and this process is repeated several times (FOGEL, 2000).

The stopping criterion in the *EAC* is the maximum number of iterations (*num_it*) that is parameterized; this criterion is usually present in evolutionary algorithms. The *EAC* also has a step for a local search execution that is made by the *k-means* algorithm (JAIN; MURTY; FLYNN, 1999; XU; WUNSCH, 2005). The local search aims to approximate the candidate solutions of clustering with good quality before the evaluation of each individual of the population. This way there is an improvement on the search for the best clustering.

Details about the encoding scheme and fitness function for EAC will be provided in the next section, when the *ocopt-aiNet* algorithm is proposed.

3 OCOPT-AINET: THE PROPOSAL OF AN OPTIMAL CLUSTERING IMMUNE ALGORITHM

The immune algorithm proposed in this work, entitled *ocopt-aiNet* (*optimal clustering opt-aiNet*), is based on the two algorithms described above: 1. *opt-aiNet* (CAS-

TRO; TIMMIS, 2002), and 2. *EAC* (HRUSCHKA; CAMPELLO; CASTRO, 2006). The *opt-aiNet* was used to define the dynamic adaptation of *ocopt-aiNet*, whilst the *EAC* was used to define the representation of individuals in the population and the genetic variation operators (HRUSCHKA; CAMPELLO; CASTRO, 2006).

Two versions of *ocopt-aiNet* are proposed in this paper. The first version (v_1), inspired by the *EAC*, has a local search performed using the *k-means* algorithm, and a second version (v_2) identical to v_1, but without the local search performed by *k-means*.

The *ocopt-aiNet* was developed with the objective of overcoming some of the drawbacks found in classical clustering algorithms, namely, the proposition of a single clustering solution at each execution, the need to automatically determine a suitable number of groups in the database, and a local search around a specific region of the space. Thus, the *ocopt-aiNet* algorithm has the following main features:

- Automatic determination of the number *k* of clusters in the database;
- Broad exploitation of the search space; and
- Generation of multiple simultaneous clusters, allowing a more exploratory and multimodal analysis of the database.

The terminology used in describing the *ocopt-aiNet* algorithm is the same as the one used to describe *opt-aiNet*:

- *Cell*: represents an individual in the population encoding a candidate solution for clustering.
- *Clones*: new generations of cells copied from the existing cells and subject to mutation.
- *Affinity*: degree of similarity between the cells, evaluated through an analysis of their coding for the cluster.
- *Suppression*: deletion of cells in the population.
- *Fitness*: value of each cell in relation to an objective function.

3.1 Encoding

The encoding scheme used in *ocopt-aiNet* is the same as the one used by *EAC*. Considering a database containing *N* objects, each cell represents a candidate solution to the clustering problem and is structured as a vector of length *N* in which each position represents the cluster to which each object belongs. If the solution has *k* clusters so the possibility of labels for an object is $\{1,...,k\}$.

The cell in Figure 1 illustrates the encoding scheme for a database containing fifteen objects and three clusters. In this case six objects {1,2,3,4,9,10} form cluster 1, cluster 2 consists of four objects {5,6,7,8}, and cluster 3 consists of five objects {11,12,13,14,15}.

Cell – [1 1 1 1 2 2 2 2 1 1 3 3 3 3 3]

Figure 1 Example of the encoding scheme for *ocopt-aiNet.*

Source: Elaborate by the authors.

Like *opt-aiNet*, the *ocopt-aiNet* has an initial population of randomly generated cells, but *ocopt-aiNet* assumes a maximum number of clusters to the positions of the vector representing each cell, a process also present in *EAC*. Assuming k initial clusters, a value ranging from 1 to k, $k > 1$, will be initially set randomly to each cell. The initial number of cells may also be set by the user or chosen randomly.

3.2 Cluster refinement

After generating the initial population, the *k-means* algorithm is used for the refinement (local search) of the clusters encoded by the cells (NALDI, 2008). The *k-means* algorithm performs a partitional clustering, in the sense that a database containing N objects is partitioned into k clusters, and positions each prototype approximately in the center of mass of each cluster. The *k-means* algorithm is summarized in Box 1.

Two parameters can be used as stopping criterion for the algorithm: a maximum number of iterations, or the maximum absolute difference between the centroid values in two consecutive iterations (HRUSCHKA; CAMPELLO; CASTRO, 2006). For *ocopt-aiNet*, only the maximum number of iterations (*max_it*) is being used as stopping criterion.

BOX 1

K-means algorithm used in *ocopt-aiNet*

Pseudocode

```
1. Assume $k$ clusters encoded in each cell;
2. While the number of iterations $N < max\_it$ do
    2.1 Calculate the centroid of the $k$ clusters;
    2.2 Encode each object with the closest centroid.
3. EndWhile.
```

Source: Elaborate by the authors.

3.3 Mutation

The *ocopt-aiNet* uses three different types of mutation operators: *exclusion*, *division* and *transformation*. The first two operators were inspired by the implementation of the evolutionary operators found in the *EAC*, and the last one was designed specifically for *ocopt-aiNet*.

For each clone *i* the mutation probability is inversely proportional to the cell fitness:

$$P(i) = 1 - f \quad (1)$$

where f is the normalized *fitness* value of the parent cell, and $P(i)$ is the mutation probability to be applied to cell i, $i = 1,\ldots, M$; M being the population size in the current iteration. Thus, the clones with parent cell that have higher *fitness* are less likely to be mutated.

Only one mutation operator can be applied to each clone at each iteration. Therefore, in case a clone mutates, the *exclusion* and *division* operators have probability $Pm_{ed} = 25\%$ to be applied, while the *transformation* operator has probability $Pm_t = 50\%$. This means that the probability of creating a new cluster or removing an existing one is half the probability of maintaining the same number of clusters and varying their structure.

The first operator (*exclusion*) is applied only in cells that have more than two clusters in their encoding, thus promoting elimination, through a random selection, of one cluster of the candidate solution. Figure 2 illustrates such case. Assuming that cluster 2 is randomly chosen to be excluded from this cell; the objects in positions 8 and 9 will have to be moved to another cluster. The new cluster(s) to which these objects will belong to is (are) chosen based on the proximity of each of these objects to the *centroid* for each remaining cluster represented by that cell.

Cell 1 – [1 1 3 4 1 3 4 $\underline{22}$ 1 3 4 4 3 3]

Figure 2 Cell representing a possible solution.

Source: Elaborate by the authors.

Figure 3 illustrates a hypothetical case of objects, with two attributes, encoded by a cell illustrated in Figure 2 and indicates the new clusters of objects 8 and 9. In Box 2 it is presented the pseudocode for the exclusion operator.

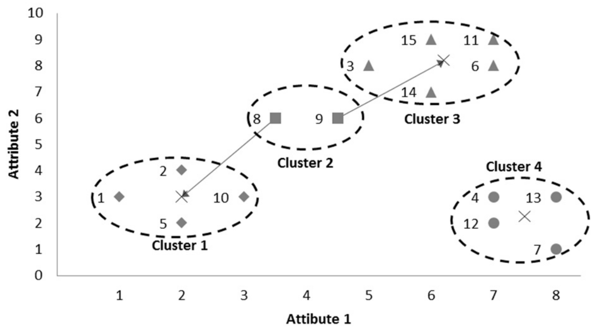

Figure 3 Exclusion operator. Clustering encoded by Cell 1, in which objects 8 and 9 originally belonging to cluster 2 are reallocated to the remaining closest clusters.

Source: Elaborate by the authors.

BOX 2

Exclusion operator

Pseudocode

```
1. Check if a cell has more than two clusters.
2. Randomly choose one cluster in the cell to be excluded.
3. Remove from the cell the label of the cluster chosen in Step 2.
4. Reallocate the objects labeled with the removed cluster to their closest clusters based on their distance to the
   centroids of the remaining clusters.
```

Source: Elaborate by the authors.

The second operator (*division*) can be applied to any cluster of a cell that has more than two objects. This operator divides a randomly chosen cluster into two new clusters, such that objects closest to the centroid of the original cluster lead to a new cluster and the objects closer to the most distant object from the centroid lead to another new cluster. As an example consider the cell illustrated in Figure 4.

Cell 2 – [113413413134433]

Figure 4 Cell used for operator of division.

Source: Elaborate by the authors.

Cluster 4 was randomly chosen for applying the division operator. In Cell 2, shown in Figure 4, cluster 4 will be divided into two clusters, one formed by objects 4, 12 and 13 (cluster 5) and another formed only by object 7 (cluster 6). These clusters with their new formations are illustrated in Figure 5. Box 3 presents the pseudocode for the division operator.

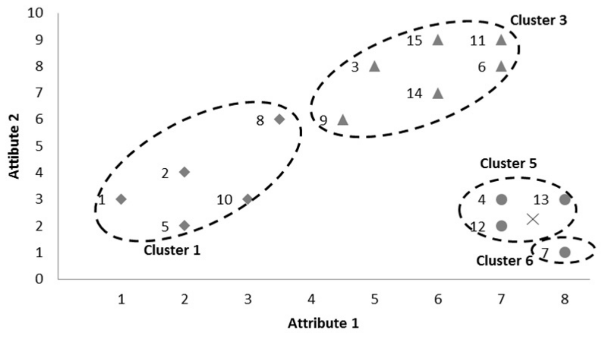

Figure 5 Division operator. Clustering encoded by Cell 2 through which the cluster 4 form two new clusters.

Source: Elaborate by the authors.

BOX 3

Division operator

```
Pseudocode
1. Randomly choose a cell cluster.
2. Check if the cluster has more than two objects.
3. Calculate the centroid of the selected cluster.
4. Calculate the similarity between each object and the centroid, and store the most distant object.
5. Reallocate the objects based on their distance to the centroid and most distant object: those objects closer to the
   centroid remain in one cluster, whilst those closer to the farer object are moved to the new cluster.
```

Source: Elaborate by the authors.

The third operator (transformation) can be applied to all positions of a cell and each position has a probability $P = 10\%$ to changing. This operator checks if the object corresponding to the position is on the right cluster. For this, it is verified the similarity of the object to all the centroids in the network. If there is a centroid of another cluster with which this object has a greater similarity than the centroid of the cluster to which it is allocated, then its label is changed to the centroid of greater similarity.

3.4 Objective function

As well as in the *EAC* algorithm, the objective function to evaluate each cell is based on the *Silhouette* criterion, which is considered a robust strategy for the prediction of optimal clusters of data (BOLSHAKOVA; AZUAJE, 2002).

To explain this concept, let us consider an object i belonging to a cluster A. So, the average dissimilarity of i to all other objects of A is denoted by $a(i)$. Now let us take into account cluster B. The average dissimilarity of i to all objects of B will be called $d(i,B)$. After computing $d(i,B)$ for all clusters $B \neq A$, the smallest one is selected, i.e., $b(i) = min\ d(i,B)$, $B \neq A$. This value represents the dissimilarity of i to its nearest neighboring cluster, and the *Silhouette*, $s(i)$, is given by:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i),\ b(i)\}} \tag{2}$$

The value $s(i) \in [-1, +1]$, the higher $s(i)$ the better the assignment of object i to a given cluster. If $s(i)$ is zero, then it is not clear whether the object should have been assigned to its current cluster or to a neighboring one. If cluster A has only one object, then $s(i)$ is not defined and is considered zero. The objective function value is the average of $s(i)$, $i = 1,\ldots, N$, and the best clustering is achieved when this function is maximized (YU; GEN, 2010; HRUSCHKA; CAMPELLO; CASTRO, 2006; BOLSHAKOVA; AZUAJE, 2002; KAUFMAN; ROUSSEEUW, 1990).

In the immune algorithm presented in this paper the *Silhouette* was changed based on the same criterion used by the *EAC* algorithm (HRUSCHKA; CAMPELLO; CASTRO, 2006): the dissimilarity of the object with a group is based on its distance to the cluster centroid, instead of its distance to all objects of the group.

Thus, the term $a(i)$ in Eq. (2) is changed to be the dissimilarity of object i to the centroid of group A. Similarly, the calculation of the term $d(i,B)$, which deals with the dissimilarity of object i to objects in group B, $B \neq A$, is changed to be the dissimilarity of object i to the centroid of cluster B, $B \neq A$. This modification allows a significant reduction of computational time, whilst maintaining the consistency with the objective function, the mutation operators (exclusion and division) and the local search (*k-means*), for they are now all based on centroids (HRUSCHKA; CAMPELLO; CASTRO, 2006). To calculate the dissimilarity between objects, the *Euclidean* distance was used.

3.5 Affinity and suppression

In *ocopt-aiNet* the affinity between the network cells has as main objective to suppress cells that have similar encodings, that is, to avoid redundancy.

In *opt-aiNet* the affinity among cells is determined by their Euclidean distance. However, in *ocopt-aiNet* the affinity among cells is calculated based on the number of objects labeled to the same cluster in the cells, even if different labels are used to encode the same clusters. It is important not to use the fitness as a synonym to affinity, because the *fitness* values of two cells may be identical or very similar, even when their encoding is very different. The affinity calculation is given by:

$$Af(a,b) = \frac{Ni}{N} \qquad (3)$$

where $Af(a,b)$ is the affinity value between cells a and b, N_i represents the total number of objects labeled in the same cluster in cells a and b, and N represents the total number of objects (length) in a cell.

Consider the network shown in Figure 6, which has four cells and represents a base with fifteen objects.

Cell 1 – [112233333334444]
Cell 2 – [221133333334444]
Cell 3 – [121233333334444]
Cell 4 – [121133333334444]

Figure 6 Network cells after the first iteration of the algorithm.

Source: Elaborate by the authors.

It can be noted that cells 1 and 2, despite having different labels for some clusters, represent the same clustering solution. Then, $Af(1,2) = 1$, meaning they are completely similar; whilst $Af(3,4) = 14/15 = 0.93$, for there is one distinct label between the two candidate solutions. The analysis of these cells is illustrated in Figure 7.

Cell 3	*Cluster I*	*Objects: 1, 3*
	Cluster II	*Objects: 2, 4*
	Cluster III	*Objects: 5, 6, 7, 8,9, 10, 11*
	Cluster IV	*Objects: 12, 13, 14, 15*
Cell 4	*Cluster I*	*Objects: 1, 3, 4*
	Cluster II	*Objects: 2*
	Cluster III	*Objects: 5, 6, 7, 8, 9, 10, 11*
	Cluster IV	*Objects: 12, 13, 14, 15*

Figure 7 Clusters generated by cells 3 and 4.

Source: Elaborate by the authors.

The suppression threshold is inversely proportional to affinity:

$$\alpha = 1 - Af \qquad (4)$$

As the affinity among cells is within the [0,1] interval, the closer α is to one, the more similar cells have to be in order for a suppression to occur.

3.6 Pseudocode

The *ocopt-aiNet* algorithm has the steps outlined in Box 4. It has two stopping criteria:

- The first criterion (Step 2.8), called the *local criterion* (δ), uses the stability of the *fitness* of cells in the population to determine if the difference between cells in the current iteration and the cells of the previous iteration is below a user defined threshold; the algorithm proceeds to the next step if the difference is less than that threshold.
- The second criterion (Step 2), called *global criterion* (Δ), aims to determine the stability of the clusters generated during a number of iterations, called a window of stability (*w*). According to this criterion, if the difference in the average number of clusters did not reach the threshold, the algorithm terminates execution, indicating that the number of clusters of cells remained stable during the window of stability.

BOX 4

ocopt-aiNet algorithm

Pseudocode

```
1. Initialize the population with $N$ random cells.
2. While average number of clusters $w$ has difference lower than $\Delta$ do
    2.1 Apply refinement clustering cells of the network (implemented using the $k$-means algorithm).
    2.2 Evaluate each cell according to their fitness.
    2.3 Apply linear normalization to the fitness.
    2.4 Generate a number $Nc$ of clones for each parent cell.
    2.5 Apply mutation proportional to the fitness of the parent cell, keeping the parent cell.
    2.6 Determine the fitness of all cells of the population.
    2.7 For each clone, select the largest fitness and calculate the average fitness of the population selected.
    2.8 If the difference between the average fitness of the population at the current iteration compared to the average fitness of the previous iteration is less than $\delta$, continue. Else, return to Step 2.
    2.9 Determine the affinity of all cells in the population. Delete cells with affinities higher than the suppression threshold ($\sigma$).
    2.10 Introduce a percentage $d$% of randomly generated cells.
3. End While.
```

Source: Elaborate by the authors.

The behavior of the algorithm can be summarized as follows. Steps 2.1 to 2.8 (local convergence): at each iteration the network cells may undergo mutation until they stabilize; Steps 2.9 to 2.10 (network suppression): after the fitness stabilization of the algorithm, the affinity among cells is calculated and the redundant ones are removed from the network. After evaluating affinity and suppression, new randomly generated cells are introduced into the network, thus contributing to a broader exploration of the search space. Step 2.1, which executes a local search with the *k-means* algorithm, was removed to create a second version of the algorithm for the experiments.

4 PERFORMANCE EVALUATION AND DISCUSSION

This section presents and discusses the experiments performed with two versions of the *ocopt-aiNet* algorithm: v_1 with the local search procedure using *k*-means, and v_2 without *k-means*. For comparative purposes we used two algorithms found in the literature and already cited in this work – the standard *k-means* and *EAC* (HRUSCHKA; CAMPELLO; CASTRO, 2006). The *EAC* was developed in order to automatically find optimal partitions of a database and the *k-means* algorithm is a classical partitional data clustering method.

4.1 Materials and methods

Three databases from the literature were used for comparative tests:

1. *Animals*: 16 objects each with 13 binary attributes; no missing values and no predefined number of clusters (HAYKIN, 1999).
2. *Ruspini*: 75 objects each with two numeric attributes and no missing value. This database has four clusters with 20, 23, 17 and 15 objects each, respectively (KAUFMAN; ROUSSEEUW, 1990).
3. *Yeast*: a bioinformatics dataset with 205 objects, each with 20 numerical attributes and no missing values. It also has four clusters with 83, 15, 93 and 14 objects each, respectively (YANG, 1993).

For each attribute of the databases a max-min normalization procedure was applied, performing a linear transformation of the original data into new values in the [0,1] domain. The algorithms were implemented using C# and the tests were performed on a PC AMD Turion X2 1.8 GHz, 2GB RAM, and operating system Windows XP Professional x64 Edition SP2. For the *ocopt-aiNet* algorithm the initialization parameters used for both versions were:

- N = 20 (initial number of cells).
- Nc = 10 (number of clones generated for each cell present in the network).
- k = 10 (initial number of clusters to be randomly generated).
- δ = 0.001 (local stopping criterion).

- $\Delta = 0.001$ (global stopping criterion).
- $w = 20$ (number of iterations of the window of stability).
- $\sigma = 0.1$ (suppression threshold).
- $d = 10\%$ (percentage of new individuals to be included at each iteration, based on the initial number of cells introduced into the network).
- *k-means*: *max_it* = 5 (maximum number of iterations, when used).

With the EAC algorithm the following parameters were used:
- $N = 20$ (number of individuals in the population).
- *num_it* = 5000 (maximum number of iterations).
- $k = 10$ (initial number of clusters to be randomly generated).
- *k-means*: *max_it* = 5 (maximum number of iterations, when used).

The *k-means* algorithm requires the definition of the number of partitions k to be used, and this number remains the same throughout the iterations. For each database 10 executions of all algorithms were performed. For the *k-means* algorithm the number k of clusters to be used will depend on the results of *EAC* and *ocopt-aiNet*, as follows. If *EAC* and *ocopt-aiNet* find an average of 10 groups with standard deviation around two, then the *k-means* was tested with values of k = 8, 9, 10, 11, 12, and, for each value of k, 10 executions were performed.

To evaluate the partitions obtained we used the *Silhouette* criterion (Eq. 2). This function indicates the quality of the partitions obtained by the simulations of the algorithms. The objective function was normalized in the [0,1] range, with higher values representing better solutions to the clustering problem.

One goal of *ocopt-aiNet* is to find multiple high quality partitions of the dataset in a single simulation. Therefore, the *adjusted rand index* – ARI (KUNCHEVA; HADJITODOROV, 2004; YEUNG; RUZZO, 2001) was used as a means to assess the diversity of the candidate solutions found by *EAC* and *ocopt-aiNet*. The index varies in the interval [–1,1], in which a value of 1 indicates a perfect agreement between partitions (maximal similarity), and negative values indicate dissimilar, diverse, candidate solutions (clustering partitions).

$$ARI = \frac{\Sigma_{i,j}\binom{n_{ij}}{2} - [\Sigma_i\binom{n_{i.}}{2}\,\Sigma_j\binom{n_{.j}}{2}]/\binom{n}{2}}{\frac{1}{2}[\Sigma_i\binom{n_{i.}}{2} + \Sigma_j\binom{n_{.j}}{2}] - [\Sigma_i\binom{n_{i.}}{2}\,\Sigma_j\binom{n_{.j}}{2}]/\binom{n}{2}} \tag{5}$$

where i and j are any two partitions, $i \neq j$; n_{ij}, $n_{i.}$, $n_{.j}$ are the number of objects in both partitions, in partition i and in partition j, respectively; and $\binom{a}{b}$ is the binomial coefficient $\frac{a!}{b!(a-b)!}$.

To display the diversity it will be used the result of:

$$Di(\mathrm{A,B}) = 1 - \mathrm{ARI(A,B)} \qquad (6)$$

where $Di(\mathrm{A,B})$ is the diversity value found for any two partitions A and B, $\mathrm{A} \neq \mathrm{B}$, and *ARI* is the value found in the calculation of the *adjusted rand index* for these two partitions.

In addition to these measures, the mean and standard deviation of the number of solutions and clusters are presented. For all simulations the time was stored and is presented in the comparison between the algorithms.

Comparisons are made by the average values of the index used. As the *EAC* and *ocopt-aiNet* algorithms not always give the same solution at the end of a simulation, the average of all simulations is used for comparison. For the *k-means*, after all the simulations for each initial *k* are performed, the average of all solutions obtained is determined. The same procedure is performed for the *ocopt-aiNet* version 1 and version 2.

4.2 Results

This section presents the results of the experiments performed with all algorithms. The *ocopt-aiNet* algorithm in its two versions is represented by v_1 (with *k-means*) and v_2 (without *k-means*). For the *Animals* and *Ruspini* databases some resulting clusters are presented to illustrate the diverse solutions found by *ocopt-aiNet*.

4.2.1 Animals

In this database, the value of parameter *k* for *k-means* varies in the interval [3,…,7]. The values for the objective function in Table 1 show that the *EAC* has a low variation, i.e., it is always trying to approximate to the same solution. The *ocopt-aiNet* v_1 and v_2 have higher variation than *EAC*, and the quality of the solutions found are better than the solutions found by *k-means*. This variation occurs due to the maintenance of various solutions and these solutions still have a good quality.

Analyzing the diversity of solutions and the number of individuals generated (Table 1), the *ocopt-aiNet* v_1 and v_2 have higher values than the other algorithms, especially the *ocopt-aiNet* v_2 that found the highest values for both. Even with a high diversity, the *ocopt-aiNet* v_2 found solutions of a good quality. The *EAC* presented the lowest number of individuals, generating only two individuals in all executions, reminding that this database does not have a pre-defined number of clusters.

The *ocopt-aiNet* v_2 also has the highest number of clusters generated by the individuals (Table 1). This is due to the fact that the algorithm tends to keep the individuals diversity and does not have an internal method to refine clusters (such as a local search in *ocopt-aiNet* v_1 and *EAC*). However, the number of clusters is almost constant and similar to the best value found by *EAC*, suggesting that this may be the best way to cluster this database.

TABLE 1

Objective function, diversity of solutions, number of individuals, number of clusters and computational time for the animals database

Algorithms	Objective function				Diversity of solutions			
	Max	*Mean*	*Min*	*SD*	*Max*	*Mean*	*Min*	*SD*
ocopt-aiNet v_1	0.775	0.757	0.740	0.012	0.510	0.441	0.356	0.061
ocopt-aiNet v_2	0.768	0.758	0.749	0.004	0.646	0.622	0.603	0.011
EAC	0.807	0.804	0.799	0.003	0.549	0.549	0.549	---
k-means	0.759	0.742	0.730	0.012	0.593	0.480	0.296	0.114
Algorithms	**Number of individuals**				**Number of clusters**			
	Max	*Mean*	*Min*	*SD*	*Max*	*Mean*	*Min*	*SD*
ocopt-aiNet v_1	6.00	4.60	3.00	0.966	4.000	3.848	3.600	0.143
ocopt-aiNet v_2	51.00	44.40	34.00	5.274	6.591	6.490	6.291	0.117
EAC	2.00	2.00	2.00	---	6.000	4.800	2.000	1.932
k-mean	---	---	---	---	4.500	3.940	3.000	0.680
Algorithms	**Computational time (mm:ss)**							
	Max	*Mean*	*Min*	*SD*				
ocopt-aiNet v_1	00:54	00:24	00:01	00:19				
ocopt-aiNet v_2	01:07	00:39	00:19	00:17				
EAC	01:13	00:42	00:34	00:13				
k-means	00:00	00:00	00:00	---				

Source: Elaborate by the authors.

The computational time (Table 1) was the lowest for *k-means*, as expected, because it is the simplest algorithm of all evaluated. The *EAC* and *ocopt-aiNet* v_2 had similar times. In the *ocopt-aiNet* v_1 the stability was found quicker than in v_2, because it has a lower number of individuals.

Table 2 illustrates the clusters found by *ocopt-aiNet* v_2 in the simulation with best results. The table presents five solutions, one in each column, divided by the clusters generated ($C_1,\ldots,C_k$; where k is the number of cluster in the solution), and the last row has the objective function value. By analyzing the objects' attributes it is possible to observe that animals in a same cluster have similar characteristics, so the algorithm was capable of finding more than one way to cluster the database. It is also very important to note that although very different in granularity, the different partitions proposed for the dataset have practically the same objective function value. This is an indication that this problem is highly multimodal with various different peaks of the objective function with similar fitness values.

TABLE 2

Clustering of database Animals by *ocopt-aiNet* v_2

1	2	3	4	5
C_1	**C_1**	**C_1**	**C_1**	**C_1**
Pigeon	Pigeon	Pigeon	Pigeon	Pigeon
Chicken	Chicken	Chicken	Goose	Chicken
Duck	Duck	Duck	C_2	C_2
Goose	Goose	Goose	Chicken	Duck
Owl	Owl	Owl	Duck	Goose
Hawk	Hawk	Hawk	C_3	C_3
Eagle	Eagle	Eagle	Owl	Owl
C_2	C_2	C_2	Hawk	Hawk
Fox	Fox	Fox	C_4	C_4
Dog	Dog	Cat	Eagle	Eagle
Wolf	Wolf	C_3	C_5	C_5
Cat	Cat	Dog	Fox	Fox
Tiger	C_3	Wolf	Cat	Cat
Lion	Tiger	C_4	C_6	C_6
Horse	Lion	Tiger	Dog	Dog
Zebra	C_4	Lion	Wolf	Wolf
Cow	Horse	C_5	C_7	C_7
	Zebra	Horse	Tiger	Tiger
	C_5	Zebra	Lion	Lion
	Cow	C_6	C_8	C_8
		Cow	Horse	Horse
			Zebra	Zebra
			C_9	C_9
			Cow	Cow
0.79921	**0.79645**	**0.80709**	**0.80001**	**0.80001**

Source: Elaborate by the authors.

4.2.2 Ruspini

In this database, the value of parameter *k* for *k-means* varies in the interval [4,…,6]. The objective function values, in Table 3, show that the *EAC* algorithm tends to always find the same solution As the *k-means* algorithm depends on the initial condition, sometimes the centroids do not present the correct clustering. The *ocopt-aiNet* v_1 and v_2 algorithms also always found the correct clustering in all simulations. The absence of local search in *ocopt-aiNetv*$_2$promoted a greater exploration and some solutions with lower quality, but kept the average closer to the other algorithms, showing that the solutions still have good quality.

Analyzing the diversity of solutions and the number of individuals generated (Table 3), the *ocopt-aiNetv*$_1$ and v_2 present higher values due to the maintenance of multiple high-quality solutions. The *k-means* algorithm has some diversity in the solutions but still lower than that of *ocopt-aiNet*.

The number of clusters presented in Table 3, show that the *ocopt-aiNet* algorithms (v_1 and v_2) perform a more exploratory search than the other algorithms. The *k-means* presented a little variation because of the interval for parameter *k*.

TABLE 3

Objective function, diversity of solutions, number of individuals, number of clusters and computational time for the Ruspini database

Algorithms	Objective function				Diversity of solutions			
	Max	***Mean***	***Min***	***SD***	***Max***	***Mean***	***Min***	***SD***
ocopt-aiNet v_1	0.859	0.849	0.832	0.008	0.414	0.311	0.352	0.077
ocopt-aiNet v_2	0.841	0.821	0.778	0.020	0.497	0.365	0.288	0.063
EAC	0.908	0.908	0.908	---	0.000	0.000	0.000	---
k-means	0.872	0.865	0.854	0.009	0.267	0.215	0.188	0.044

Algorithms	Number of individuals				Number of clusters			
	Max	***Mean***	***Min***	***SD***	***Max***	***Mean***	***Min***	***SD***
ocopt-aiNet v_1	5.00	3.80	3.00	0.632	6.333	5.548	4.750	0.509
ocopt-aiNet v_2	18.00	12.20	7.00	3.359	7.071	6.678	6.000	0.307
EAC	1.00	1.00	1.00	---	4.000	4.000	4.000	---
k-mean	---	---	---	---	5.600	4.700	3.600	1.014

Algorithms	Computational time (mm:ss)			
	Max	***Mean***	***Min***	***SD***
ocopt-aiNet v_1	01:16	00:26	00:05	00:22
ocopt-aiNet v_2	05:34	01:57	00:08	02:00
EAC	02:31	02:09	01:41	00:19
k-means	00:00	00:00	00:00	---

Source: Elaborate by the authors.

In this database, the computational time (Table 3) for *ocopt-aiNetv*$_1$ and v_2 algorithms presented high variations. Version v_1 stabilized more quickly than v_2 because it is favored by the presence of *k-means* (local search). Comparing with the *EAC*, the *ocopt-aiNet* versions obtained better average time to global stability showing that the algorithms tend to quickly find good quality solutions.

Figure 8 illustrates the three different clusterings (each cluster represented by a different geometric shape and gray level) with highest objective function values for the *ocopt-aiNet* algorithms. It can be noted that the algorithms could find more refined clusters than the natural ones available in the dataset. This can be useful in databases in which the number of clusters is not known. Again, substantially different partitions lead to quite similar fitness values.

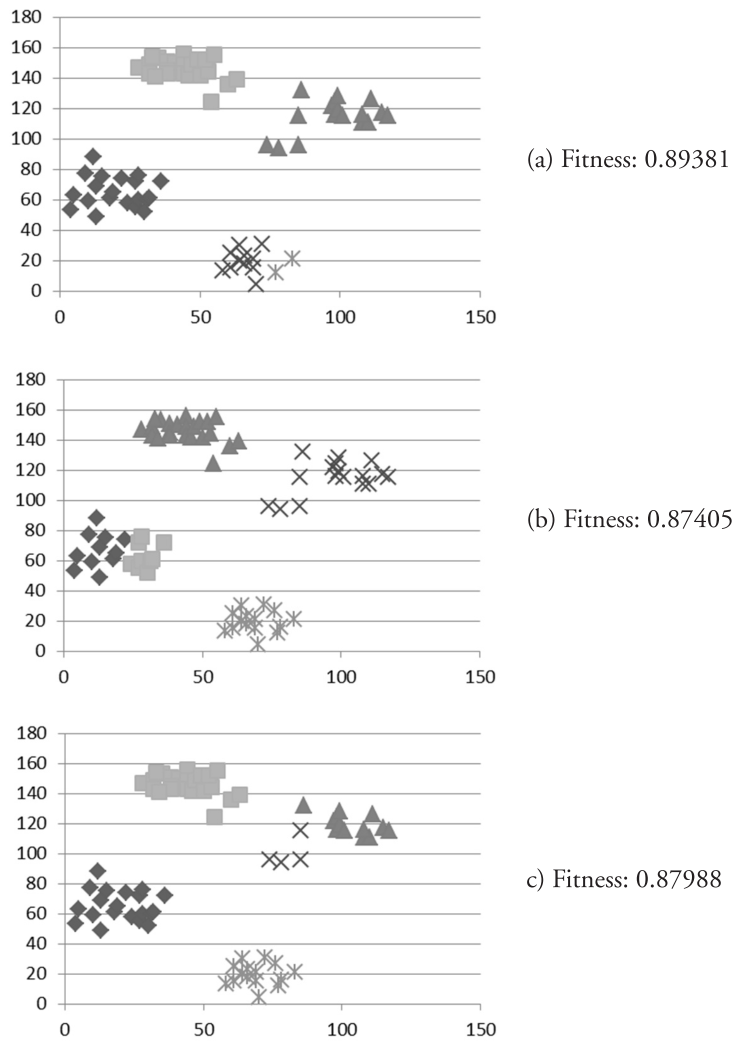

Figure 8 *ocopt-aiNet* clusterings for the Ruspini database.

Source: Elaborate by the authors.

4.2.3 Yeast

For the Yeast dataset the *k-means* algorithm maintained good quality solutions and, on average, better than the ones found by *ocopt-aiNet* v_1 and v_2 (Table 4). The objective function values for *ocopt-aiNet* were almost the same in all simulations, as can be noted by the low standard deviations.

The *ocopt-aiNet* v_1 and v_2 algorithms presented higher diversity of solutions than *EAC* and *k-means* (Table 4). The *EAC* algorithm presented only one individual for all simulations, so there is no diversity at all. The *ocopt-aiNet* v_2 found the highest number of individuals with highest diversity.

By the analysis of the number of clusters (Table 4) the *ocopt-aiNet* v_2 algorithm performed a broader search than the other algorithms. *ocopt-aiNet* v_1 and *k-means* presented similar numbers of clusters, suggesting that the best clustering found by *k-means* made *ocopt-aiNet* v_1 keep its cells closer to this number of clusters. The *EAC* algorithm presented only one clustering without variation.

As the *Yeast* database has four known clusters, no algorithm was capable of finding the correct clustering for this database.

TABLE 4

Objective function, diversity of solutions, number of individuals, number of clusters and computational time for Yeast database

Algorithms	Objective function				Diversity of solutions			
	Max	*Mean*	*Min*	*SD*	*Max*	*Mean*	*Min*	*SD*
ocopt-aiNet v_1	0.782	0.762	0.742	0.011	0.559	0.380	0.255	0.104
ocopt-aiNet v_2	0.759	0.737	0.704	0.017	0.519	0.397	0.269	0.078
EAC	0.841	0.841	0.841	--	0.000	0.000	0.000	---
k-means	0.839	0.805	0.771	0.026	0.246	0.146	0.056	0.077

Algorithms	Number of individuals				Number of clusters			
	Max	*Mean*	*Min*	*SD*	*Max*	*Mean*	*Min*	*SD*
ocopt-aiNet v_1	8.00	6.10	4.00	1.197	5.50	5.20	4.80	0.224
ocopt-aiNet v_2	23.00	17.20	14.00	4.442	7.95	7.48	7.11	0.306
EAC	1.00	1.00	1.00	---	3.00	3.00	3.00	---
k-mean	---	---	---	---	5.40	4.22	2.80	0.998

Algorithms	Computational time (mm:ss)			
	Max	*Mean*	*Min*	*SD*
ocopt-aiNet v_1	46:07	18:58	04:12	14:28
ocopt-aiNet v_2	209:25	106:09	09:47	65:01
EAC	30:10	29:35	29:07	00:21
k-means	00:00	00:00	00:00	---

Source: Elaborate by the authors.

The computational time (Table 4) for *ocopt-aiNet* v_2 was higher in relation to the other algorithms, and presented a high variation among the simulations. The *ocopt-aiNet* v_1 presented a processing time closer to the *EAC* algorithm, reminding that the *EAC* has as stopping criterion the number of iterations, keeping the variation low between the simulations. The *k-means* algorithm, as in the other analyses, had the lowest computational time, reaching the stability very quickly.

4.3 General discussion

Finding diversity in a database is an important ability to support decision making (HAN; KAMBER, 2000), thus allowing a broader analysis of the existing relationships among the objects of a database. The *ocopt-aiNet* v_2 algorithm, after the comparative results presented for the three databases, showed to be the most effective in generating and maintaining the diversity of solutions. Besides, the individuals found by this algorithm are of good quality. In some cases, the *k-means* algorithm presented a better average quality (*Yeast* database), but the diversity found by *k-means* was lower than that of *ocopt-aiNet* v_2.

For databases that do not have a predefined number of clusters (e.g., *Animals* database), the *ocopt-aiNet* v_2 algorithm presented a large number of individuals as candidate solutions for the clustering, differently from the other algorithms that presented a small number more frequently.

In contrast to the diversity, the *ocopt-aiNet* algorithm had a higher computational cost. However, it is important to emphasize that it is the price paid by the automatic search for the number of clusters, and by the maintenance of population diversity. It is evident that there is a balance between multimodality, dynamic search of the parameterization and computational efficiency. The decision of what will be privileged in an experiment will depend on the goal of the problem.

5 CONCLUSION

This work proposed two versions of a new immune algorithm for multimodal data clustering, called *ocopt-aiNet*. Both have an automatic determination of the number of clusters, maintenance of multiple high-quality solutions, and broad exploration of the search space. The difference between them is the presence of a local search procedure that promotes a faster convergence at the expense of a loss in diversity.

By the results presented, it was observed that for databases with a large number of objects, the proposed algorithm in the version without local search (v_2) presented a high computational cost when compared with its other version and algorithms. This results from the difficulty encountered by the algorithm to stabilize the solutions, but, at the same time, makes the algorithm find a larger number of solutions and, in some experiments, with better quality and diversity than all other algorithms.

For databases that allow multimodal high-quality solutions, such as the *Animals* and *Ruspini* databases, the experiments showed that the proposed algorithm, especially the version without local search (v_2), was able to find a high number of good-quality solutions, benefiting the analysis of possible clustering solutions. For such databases, in which the number of clusters is unknown, and different clusterings are possible, the proposed algorithm appears to be effective.

Finally, for the cases where there is a need to find multiple solutions in a same database, the *ocopt-aiNet* algorithm has relevant results. In cases where the goal is to find one optimal clustering, this algorithm does not seem to be efficient due to the high computational cost.

As further research it is possible to detach a sensitivity analysis of the proposed algorithm to its input parameters, the application to a much broader set of databases and a more thorough comparison with other approaches from the literature.

OCOPT-AINET: UM SISTEMA IMUNOLÓGICO ARTIFICIAL PARA AGRUPAMENTO ÓTIMO DE DADOS

Resumo

Agrupamento de dados é uma importante tarefa da mineração de dados. Boa parte dos algoritmos pode agrupar os objetos de forma simples e eficiente, mas possui inconvenientes como a forma de obter o número ótimo de grupos e a possibilidade de ficar preso em ótimos locais. Para tentar diminuir esses inconvenientes, este artigo propõe um novo algoritmo imunológico para agrupamento de dados baseado em sistemas imunológicos artificiais. Esse algoritmo é caracterizado pela geração de múltiplas soluções simultâneas de boa qualidade no que tange ao número de grupos e a uma função de custo que avalia explicitamente a qualidade desses grupos, minimizando o inconveniente de ficar preso em ótimos locais.

Palavras-chave: Sistemas imunológicos artificiais, agrupamento de dados, ocopt-aiNet.

REFERENCES

BOLSHAKOVA, N.; AZUAJE, F. Cluster validation techniques for genome expression data. Dublin, 2002. Disponível em: <https://www.cs.tcd.ie/publications/tech-reports/reports.02/TCD-CS-2002-33.pdf>. Acesso em: 19 out. 2008.

CASTRO, L. N. de; TIMMIS, J. An artificial immune network for multimodal function optimization. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION, 2., 2002, Hawaii. *Proceedings*... Hawaii, 2002. v. 1, p. 699-674.

CASTRO, L. N. de; TIMMIS, J. I. Artificial immune systems as a novel soft computing paradigm. *Soft Computing*, v. 7, n. 8, p. 526-544, 2003.

CASTRO, L. N. de; VON ZUBEN, F. J. aiNet: an artificial immune network for data analysis. In: ABBASS, H. A.; SARKER, R. A.; NEWTON, C. S. (Ed.). *Data mining*: a heuristic approach. Hershey: Idea Group Publishing, 2001. p. 231-259.

DASGUPTA, D.; JI, Z.; GONZALEZ, F. Artificial immune system (AIS) research in the last five years. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 3., 2003, Canberra. *Proceedings*... Canberra: IEEE, 2003. p. 123-130.

FOGEL, D. B. Evolutionary computing. *IEEE Sprectrum*, Piscataway, v. 37, n. 2, p. 26-32, Feb. 2000.

HAN, J.; KAMBER, M. *Data mining*: concepts and techniques. San Francisco: Morgan Kaufman, 2000.

HAYKIN, S. *Neural networks*: a comprehensive foundation. 2. ed. Upper Saddle River: Prentice Hall, 1999.

HRUSCHKA, E. R.; CAMPELLO, R. J. G. B.; CASTRO, L. N. de. Evolving clusters in gene-expression data. *Information Sciences*, v. 176, n. 13, p. 1898-1927, 2006.

HRUSCHKA, E. R. et al. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, v. 39, n. 2, p. 133-155, 2009.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM Computing Surveys*, v. 31, n. 3, p. 264-323, 1999.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data*: an introduction to cluster analysis. New York: John Wiley & Sons, 1990.

KUNCHEVA, L.; HADJITODOROV, S. Using diversity in cluster ensembles. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2004.

NALDI, M. C. *Investigation of techniques of evolutionary computation problems in data clustering*. 2008. Thesis (Ph.D. in Computer Sciences and Computational Mathematics)–Universidade de São Paulo, São Carlos, 2008.

SHEIKH, R. H.; RAGHUWANSHI, M. M.; JAISWAL, A. N. Genetic algorithm based clustering: a survey. In: INTERNATIONAL CONFERENCE ON EMERGING TRENDS IN ENGINEERING AND TECHNOLOGY, 1., 2008, Nagpur, Washington. Nagpur, Washington: IEEE, 2008. p. 314-318.

XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, v. 16, n. 3, p. 645-678, 2005.

YANG, M. S. A survey of fuzzy clustering. *Mathematical and Computer Modeling*, v. 18, n. 11, p. 1-16, 1993.

YEUNG, K. Y.; RUZZO, W. Details of the adjusted rand index and clustering algorithms. Supplement to the paper "an experimental study on principal component analysis for clustering gene expression data". *Bioinformatics*, v. 17, n. 9, p. 763-774, 2001.

YU, X.; GEN, M. *Introduction to evolutionary algorithms*. London: Springer-Verlag, 2010.

Contato

Leandro Nunes de Castro
e-mail: lnunes@mackenzie.br