Research Article

Bacterial Colony Algorithms for Association Rule Mining in Static and Stream Data

Danilo S. da Cunha ¹, **Rafael S. Xavier**,¹ **Daniel G. Ferrari**,¹
Fabício G. Vilasbôas,^{1,2} and **Leandro N. de Castro** ¹

¹Natural Computing and Machine Learning Laboratory (LCoN), Mackenzie Presbyterian University, São Paulo 01302-907, Brazil

²AXONDATA Analytics Technology LLA, São Paulo, Brazil

Correspondence should be addressed to Danilo S. da Cunha; danilocunha85@gmail.com

Received 9 May 2018; Revised 25 September 2018; Accepted 9 October 2018; Published 11 November 2018

Guest Editor: Eric Monfroy

Copyright © 2018 Danilo S. da Cunha et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Bacterial colonies perform a cooperative and distributed exploration of the environmental resources by using their quorum-sensing mechanisms. This paper describes how bacterial colony networks and their skills to explore resources can be used as tools for mining association rules in static and stream data. A new algorithm is designed to maintain diverse solutions to the problems at hand, and its performance is compared to that of other well-known bacteria, genetic, and immune-inspired algorithms: Bacterial Foraging Optimization (BFO), a Genetic Algorithm (GA), and the Clonal Selection Algorithm (CLONALG). Taking into account the superior performance of our approach in static data, we applied the algorithms to dynamic environments by converting static into flow data via a stream data model named sliding-window. We also provide some notes on the running time of the proposed algorithm using different hardware and software architectures.

1. Introduction

Bacterial colonies can be seen as complex adaptive systems that perform distributed information processing to solve complex problems, such as food acquisition, swarming mobility, and biofilm formation, among others. They use a collaborative system of chemical signals to explore the resources of a given environment and coordinate their social and behavioural tasks [1]. Bacteria can be found in distinct environments, ranging from hostile to more hospitable ones by applying different kinds of survival strategies to process self and environmental stimuli [2].

The collective and collaborative activities carried out by a bacterial colony are classified as a type of collective intelligence [3], where each bacterium is able to sense itself and the environment and maintain communication with other bacteria in the colony to perform its coordinated tasks. This enables the colony to acquire information about the environment and its changes. Thus, a colony can be seen as an adaptive computational system that processes information on different levels, independently of environmental changes

[4]. Some important computational properties and collective behaviours of bacteria colonies are shown in [4].

This paper presents an algorithm inspired by the exploratory behaviour of environmental resources by a colony of bacteria, named BaCARO-II, extended from [5, 6], for mining association rules of items in transactional databases and introduces the necessary modifications so that it can be applied to data streams. As an outcome of the modifications, the new bacteria algorithm is able to avoid the genic conversion problem discussed in [7].

The bacterial colony algorithm is compared to other bio-inspired heuristics, more specifically the Bacterial Foraging Optimization (BFO) [8], a Genetic Algorithm (GA) [9], and the Clonal Selection Algorithm (CLONALG) [10], which were adapted to perform association rule mining of static and stream data. The following performance measures are accounted for: *support* (S), *confidence* (C), *interestingness* (I), *number of rules* (U), and *processing time* (P).

The paper is an extension of [11] and it is organized as follows. Section 2 provides some theoretical background on association rule mining and Section 3 a review of data

stream processing models. Section 4 provides the biological foundations of bacterial colonies and Section 5 presents an overview of bacterial algorithms. Section 6 introduces two bacterial algorithms applied to association rule mining in static and dynamic environments. Section 7 shows the experimental results and, finally, the final considerations and future works are provided in Section 8.

The abbreviations used for the algorithms in this research are as follows:

BaCARO-II: Bacterial Colony Association Rule Optimization-II

BFO: Bacterial Foraging Optimization

CLONALG: Clonal Selection Algorithm

GA: Genetic Algorithm

sBaCARO-II: Stream Bacterial Colony Association Rule Optimization-II

sBFO: Stream Bacterial Foraging Optimization Algorithm

sCLONALG: Stream Clonal Selection Algorithm

sGA: Stream Genetic Algorithm

2. On Association Rule Mining and Data Streams

This section provides a brief review of the two main concepts covered in this paper: association rule mining and data streams.

2.1. Association Rule Mining. Originally known as market-basket analysis, mining association rules is one of the main data mining tasks. It is a descriptive task, which uses unsupervised learning and focuses on the identification of associations between items that occur together in a dataset [12–15]. A *transaction* is a set of items that occur together. In the scenario described in the original market-basket analysis, items in a transaction are those that are acquired together by an end user [14, 15]. An association rule is as follows:

$$A \longrightarrow C \quad (1)$$

where A and C are itemsets of products selected by a consumer.

The first set A is called the antecedent and the other one C is called the consequent of the association rule. The intersection between these two sets is empty ($A \cap C = \emptyset$), because it is redundant for an item to imply itself. The rule means that the presence of (all items in) A in a transaction implies the presence of (all items in) C in the same transaction with some associated probability [13, 15].

Given a set of transactions T , it is interesting to generate all rules that satisfy two types of constraints:

- (i) *Syntactic constraints:* the number of items that appear in a rule is limited.
- (ii) *Support constraints:* involving delimitations in the number of transactions in T that support the rule,

FIGURE 1: Number N of possible rules, given number d of items.

with support, usually an input parameter, being defined as the number of transactions in T that contain A and C simultaneously.

The problem with the previous definition is that the number N of possible association rules, given a number d of items, grows exponentially, and the problem is placed within the NP-complete set [12, 13, 15]:

$$N = 3^d - 2^{d+1} + 1 \quad (2)$$

To illustrate how this scales, Figure 1 shows the value of N for growing values of d .

Therefore, it is not computationally feasible to generate all rules for fairly large datasets in a reasonable time. Thus, it is compulsory to somehow prune the association rules built before trying to analyse their real usefulness.

Measures of Interest. The *Confidence* and *Support*, proposed in [12, 13], are the most studied and applied measures of interest in the association rule mining literature. The support of an association rule is a measure of its relative frequency in the set of all transactions:

$$\text{Support}(A \longrightarrow C) = \text{Supp}(A \longrightarrow C) = P(A \cup C) \quad (3)$$

On the other hand, the confidence of a rule is a measure of its satisfiability or strength when its antecedent is found in T , that is to say, from all the occurrences of A , how often C also occurs in the base:

$$\text{Confidence}(A \longrightarrow C) = \text{Conf}(A \longrightarrow C) = P(A | C) \quad (4)$$

While confidence is a measure of the strength of a rule, the support corresponds to its statistical significance over the database. The interestingness of a rule, $I(A \longrightarrow C)$, is calculated as follows [14]:

$$I(A \longrightarrow C) = \left(\frac{|A \cup C|}{|A|} \right) * \left(\frac{|A \cup C|}{|C|} \right) * \left(1 - \left(\frac{|A \cup C|}{T} \right) \right) \quad (5)$$

where A and C are defined as previously and T is the number of transactions in the database. This measure of interest,

FIGURE 2: Main stream processing models.

differently from the support, looks for low frequency rules in the database.

The Apriori Algorithm. The most well-known algorithm for association rule mining is called Apriori [13] and has the following main steps:

- (i) **Generate frequent itemsets:** a set of frequent items is the one whose support is greater than or equal to a minimum support threshold (*minsup*).
- (ii) **Generate reliable association rules:** the reliable association rules are those with a confidence value equal to or greater than a minimum confidence value (*minconf*).

A set of items of length k , i.e., with k items, is called a k -itemset. The Apriori algorithm was named after its use of a methodology for selecting items that come before others (*a priori*) for the generation of frequent itemsets. This feature is known as closing down.

The algorithm performs multiple scans over the database. In the first step it computes the frequency of each item. After keeping those items whose frequency is equal to or greater than *minsup*, it checks if those frequent items, i_x , occur in conjunction with item i_{x+1} and together if their frequency is greater than or equal to *minconf*. At each new iteration on the data, the algorithm stores, incrementally, only those frequent items that satisfy *minsup* and *minconf*. Therefore, Apriori-based algorithms are not suitable for a data stream environment, because data can be scanned only once [16].

3. Data Streams

A sequence of objects that arrives in a timely order is named a data stream [17, 18]. Differently from traditional static data, data streams are continuous, unbounded, and of high speed and their data distribution changes with time. Data streams can be classified in two main classes: offline streams and online streams. An offline stream is characterized by regular bulk arrivals, while an online stream is characterized by real-time updated data that come one followed by the other in time. Unlike offline data streams, bulk data processing is not possible for online stream data [19]. As the number of applications over data streams grows rapidly, there is an increasing need to perform data stream mining tasks, such as classification, clustering, and association rule on stream data.

There are three major stream data processing models for rule mining [20]:

- (i) **Landmark model:** it mines all frequent itemsets over the entire log of stream data from a limited point of time, named *landmark*, to the current one. This simple model is not suitable for applications where the user is interested in the most recent information of data streams.
- (ii) **Damped model:** also named time-fading model, it finds frequent itemsets in stream data in which each transaction has a weight decrease with time. Older transactions have a smaller weight toward itemset frequencies, i.e., different weights for new and old transactions.
- (iii) **Sliding-window model:** it finds and maintains frequent itemsets in sliding-windows. Only part of the data streams within the sliding-windows are stored and processed at the time while the data flows in. The sliding-window size is defined based on the application and system resources. The result depends on recently generated transactions in the window range.

All approaches have been used in different researches on data stream mining. Selecting which kind of stream data process model to use largely depends on the application demands. The three approaches are summarized in Figure 2.

Some data stream applications involving association rule mining include estimating missing data in sensor networks [21]; predicting the frequency of Internet packet streams [22]; finding alarm incidents from streams [23]; determining frequent itemsets over online data streams [24]; and association analysis [25–27].

Open Problems in Data Stream Association Rule Mining. Despite the many applications, these tools are focused on specific areas, and none of them fully deal with the main open issues in data stream association rule mining [16]:

- (i) There is not enough time to rescan the whole database or to perform a multiscan, as in traditional data mining algorithms.
- (ii) The data stream mining method needs to adapt to the data distribution, i.e., avoid the *drifting problem* [28].
- (iii) The speed of the mining algorithm should be faster than the data arrival rate.
- (iv) Due to the stream properties, the analysis results of data streams often keep changing as well.
- (v) A mining mechanism that adapts itself to the available resources is needed.

4. Some Notes on Bacterial Colonies

Bacterial colonies have different behavioural patterns, including *foraging*, *reproduction*, *communication*, *sporulation*, and *motility* [29, 30]. They perform a distributed and parallel information processing and each bacterium is an autonomous system capable of sending, storing, processing, and interpreting information. This gives the bacterium a certain freedom to choose its response according to the messages received as part of the chemical distributed processing of information from the colony.

Bacterial communication occurs via chemical signals. The main entities around this communication are the *signalling cell*, the *target cell*, the *signal molecule*, and the *receiver protein*. The signalling cell sends the chemical signal, presented by the signal molecule, to one or more target cells. The target cells read the message contained in the signalling molecule via protein receptors and then send the message to the intracellular gel. The signalling molecule does not enter the bacteria; the responsible one for decoding and sending each message to the intercellular plasma is the receiver protein [31].

The most studied bacterial communication process in the literature is *quorum-sensing*, which depends on the concentration of a diffusible molecule called *autoinducer* [32, 33], and works only in a high density colony. The concentration of autoinducers increases in the environment with the growth of the number of cells that produce them, thus promoting the activation or suppression of gene expression that are responsible for generating certain behaviours in bacteria. Quorum-sensing works as a micro and macro communication mechanism. In the *intracellular communication* network, a bacterium analyses and interprets the data read from the environment. The macro level information processing is represented by the biochemical interactions of the colony, which correspond to the *extracellular communication*.

The motion patterns, named *taxes*, that the bacteria generate in the presence of chemical attractants and repellents are called *chemotaxis*. The bacteria movement can be done by *swimming*, which means moving in the same direction, and if a bacterium performs successive swimming steps, we say it is performing a *running* step, and, finally, if it is moving in a random direction we say it is *tumbling*. Swimming and tumbling (chemotactic behaviour) are individual and stochastic responses that result in emergent global responses, such as swarming.

Reproduction in bacteria is performed after some chemotaxis steps. The bacteria fitness is used to select those who will die, and the survivors are divided into two new bacteria placed in the same direction. In other words, the survivors are cloned via asexual reproduction, and the clones stay in the same region as their parents.

5. Bacterial Colony Algorithms: BFO and BaCARO-II

There is currently a number of bacteria-inspired algorithms. The pioneer proposal was called Bacterial Chemotaxis

Algorithm (BCA) [34] and bacterial foraging behaviours have been used as inspiration for the design of other algorithms, such as the Bacterial Foraging Optimization (BFO) Algorithm [8], Bacterial Colony Optimization (BCO) [35], and Bacterial Colony Association Rule Optimization (BaCARO) [5, 6]. This section describes BFO, which is one of the most well-known proposals in the literature, and a version of our approach, named BaCARO-II. The nomenclature of the parameters used by the algorithms is as follows:

P: population of candidate solutions

Bac_{num} : number of bacteria in a population

N_{ed} : number of elimination and dispersal steps

N_{re} : number of reproduction steps

N_c : number of chemotactic steps

N_s : number of swim steps

P_{ed} : probability of elimination-dispersal

ic_{Prob} : probability of intracellular communication

ec_{Prob} : probability of extracellular communication

P_{cha} : probability of changing information

$Size_{net}$: extracellular network size

5.1. The Bacterial Foraging Optimization Algorithm: BFO. The Bacterial Foraging Optimization (BFO) algorithm simulates the foraging strategy of *Escherichia Coli* and was originally designed to solve optimization problems in continuous environments. It takes inspiration in the following bio-inspired mechanisms [8, 36]: chemotaxis, reproduction, elimination, and dispersion.

Algorithm 1 summarizes the main steps of the BFO algorithm for solving a minimization task. It starts by initializing all the input parameters: a colony **P** with Bac_{num} bacteria of the same dimension as the problem to be solved; number of elimination and dispersal steps (N_{ed}); number of reproduction steps (N_{re}); number of chemotactic steps (N_c); number of swim steps (N_s); the elimination-dispersal probability (P_{ed}); and number of bacteria to be selected for reproduction (S_r).

The algorithm first applies chemotaxis and reproduction until their thresholds are reached and then follows with elimination-dispersal. During reproduction a bacterium is cloned (duplicated) with no mutation. During chemotaxis, the health (fitness) of each bacterium is assessed and a number S_r of the healthiest ones are cloned, while the others are removed from the population. Bacteria are then allowed to swim for a number of swim steps (N_s), moving to different locations. If the new location results in improved (healthier) bacteria, then they keep swimming in the same direction; otherwise they tumble, exploring other regions of the search space. Finally, bacteria can survive or be removed from the population with probability P_{ed} . Whenever a bacterium is eliminated, another one is generated in a random position (disperse).

BFO is the bacterial-inspired algorithm more extensively applied to solve problems in different areas [37, 38], such


```

procedure [P] = BFO (  $Bac_{num}$  ,  $N_{ed}$  ,  $N_{re}$  ,  $N_c$  ,  $N_s$  ,  $P_{ed}$  ,  $S_r$  )
initialize P (  $Bac_{num}$  )
for l=0 to  $N_{ed}$  do // Elimination-dispersal loop
    for k=0 to  $N_{re}$  do // Reproduction loop
        for j=0 to  $N_c$  do // Chemotaxis loop
            Apply chemotaxis
            foreach Bacterium in P do
                if Fitness (Bacterium)  $\geq$  Fitness (  $Bac_{best}$  ) then
                     $Bac_{best} \leftarrow$  Bacterium
                end if
            end foreach
        end for // Chemotaxis
         $P_{selected} \leftarrow$  SortByCellFitness ( P ,  $S_r$  )
        P = Clone (  $P_{selected}$  )
    end for // Reproduction
    foreach Bacterium in Population do
        if Random()  $\leq$   $P_{ed}$  then
            Bacterium  $\leftarrow$  BacteriumAtRandLocation()
        end if
    end foreach
end for // Elimination-dispersal
return  $Bac_{best}$ 
end procedure

```

ALGORITHM 1: Pseudocode of the BFO algorithm [36].

as global optimization [39], engineering design [40], power system [41–43], optimal design [44], network planning [45], and data analysis [46–48].

5.2. The Bacterial Colony Association Rule Optimization Algorithm: BaCARO-II. The algorithm named Bacterial Colony Association Rule Optimization-II (BaCARO-II) is inspired by the biological processes of intra- and extracellular communication networks of bacterial colonies, as well as quorum-sensing, chemotaxis, and bacterial dispersion [1, 49]. In BaCARO-II, intracellular communication [50] is used to search better gene rearrangements so that bacteria present a higher fitness, and extracellular communication is used to coordinate bacterial motility over the search space. Quorum-sensing is applied to evaluate the neighbourhood and use the synergy of individual and collective decisions, and chemotaxis is used to make fine adjustments during intracellular communication: if the new gene arrangement is worse than the previous one (position in the search space), it can be undone. Finally, dispersion promotes the movement of bacteria away from regions of high concentrations of bacteria.

BaCARO-II starts by initializing a random colony of size equal to the search-space dimension. The artificial colony is evaluated and each bacterium has a probability of making intracellular communication ic_{Prob} . The bacteria randomly selected to perform intracellular communication reconfigure their gene expression and if the new rearrangement is better than the previous one, the latter is adopted. The colony fitness is updated and the extracellular step begins. Each bacterium starts to perceive its neighbourhood, and those in the same region disperse to new regions. Those that are not occupying dense regions are selected with some

probability ec_{Prob} , a total of $Size_{net}$ surrounding bacteria to change information with their neighbours according to a P_{cha} value and move to the best direction. After that, fitness is computed. Finally, the colony is confronted with an environmental pressure that leads to the selection of the bacteria with highest fitness values to the next generation. The synergy of intracellular and extracellular communication results in quorum-sensing, which is the core of most bacterial algorithms. The pseudocode of BaCARO-II is summarized in Algorithm 2.

6. Bacterial Colonies in Association Rule Mining

This section describes how the different bacteria-inspired algorithms were adapted to solve association rule mining problems in static and dynamic environments. As presented in the previous section, BFO takes into account reproduction, chemotaxis (tumbling and swimming), and elimination-dispersal mechanisms. By contrast, BaCARO-II uses chemotaxis (tumbling and swimming), intra- and extracellular communication, and dispersion. These mechanisms will be presented here so that both algorithms can be applied to solve association rule mining tasks.

6.1. Encoding Scheme. Instead of initializing the agents in a real interval ($\mathbb{R}$), we randomly set them as pairs of binary values (00, 01, 10, or 11) for each vector position. A pair of bits represents each item in a transaction, where items present in the association rule are represented by a bit pair of 00 (antecedent of the rule) or 11 (consequent of the rule). Items

```

procedure [P] = BaCARO-II( ecprob , icprob , Pcha )
initialize P
t ← 1
f ← evaluate(P)
while not_stopping_criterion do
  for i=0 to Size(P) do //Intracellular communication loop
    rf ← inCellular( P , icprob , Pcha )
    f ← update( f , rf )
  end for
  for j=0 to Size(P) do //Extracellular communication loop
    PexCellularNetworks ← exCellular( ecprob )
    Foreach Pbacterium in each PexCellularNetworks do
      if bacterialDensity( PexCellularNetworks ) == true then //Quorum-sensing
        P ← disperse( Pbacterium ) //Dispersion
      else
        Foreach Pbacterium in PexCellularNetworks do
          PexCellularNetworks ← MoveToBestDirectionInExtracellularNetwork( Pbacterium )
        end Foreach
      end Foreach
    end for
    P ← PexCellularNetworks
    f ← evaluate(P)
    P ← select( P , f )
    t ← t+1
  end while
end procedure

```

ALGORITHM 2: Pseudocode of BaCARO-II algorithm.

A	B	C	D	E	F	G	H
11	00	01	10	00	11	10	01

FIGURE 3: An artificial bacterium codifying an association rule.

out of a rule are composed of the other combinations: 01 or 10. Figure 3 illustrates an artificial bacterium encoding the following rule: $B \wedge E \rightarrow A \wedge F$.

6.2. Reproduction. The surviving bacteria are cloned without mutation.

6.3. Chemotaxis: Swim and Tumble. Another modification to mine association rules was made in the chemotactic behaviors. A rule of size $n - 1$ is more probable than a rule of size n . The tumbles were implemented by randomly choosing a rule part (antecedent or consequent) to be shortened and removing an element from this part. If after the tumble the bacterium adaptation level (fitness) increases, it starts to run (applying swim steps) by removing items from the same part until its size is equal to 1 or the number of swim steps (user-defined parameter) is reached, as illustrated in Figure 4. Note that, in terms of chromosomes, the bacteria maintain the same length after swim and tumbling; what changes is only the number of items in the encoded rules.

On the other hand, if after tumbling the bacterium maintains its adaptation value (fitness) the chemotactic behavior is finalized, as illustrated in Figure 5.

6.4. Elimination-Dispersal Mechanisms. This step has two parts:

- (1) Elimination: removal of some bacteria from the colony based on their fitness (adaptability).
- (2) Dispersal: randomly changing the positions of the bacteria in the search space.

6.5. Intracellular Communication. In this step each bacterium has an associated probability of performing internal communication. The parts that make up a rule are identified as exchanging structures and the items of these structures may assume a new position in the rule, that is, a new gene expression, as illustrated in Figure 6.

6.6. Extracellular Communication. Extracellular communication is used to coordinate bacterial motility as a collective behaviour over the search space by sharing information in a chemical network. The chemical network is used to control the range of information into a part of the colony, a group. In our model, the information shared by the bacterium with higher fitness is considered by the others as a reference to move around the search space. In a higher density group, the collective behaviour adopted is to disperse to new regions.

6.7. Evaluation Function. The *evaluation*, *fitness*, or *objective function* should reflect the relevance of the measures to be optimized, exhibit regularities over the space defined by the chosen representation, and provide enough information

FIGURE 4: Tumbling followed by a swim step. In this example tumble selected the consequent part to shorten and swim was performed until the bacterium had a single item in the consequent part of the rule.

FIGURE 5: Chemotactic behaviour finished by a tumbling step.

to drive the environmental pressure of a population-based search algorithm [51]. The measures of interest often used in Evolutionary Algorithms and Artificial Immune Systems to compute fitness values are based on those employed for classification rule mining, with some slight modifications.

Confidence and support were used in [52–54] to define the fitness function as

$$Fitness_1(A \rightarrow C) = \left(\frac{w_1 * Supp(A \rightarrow C)}{minSupp} \right) * \left(\frac{w_2 * Conf(A \rightarrow C)}{minConf} \right) \quad (6)$$

where $w_1 = w_2 = 0.5$ and $w_1 + w_2 = 1$ and $minSupp$ and $minConf$ are, respectively, the user-defined minimum threshold values for support and confidence. Another fitness function present in the association rule mining literature is

$$Fitness_2(A \rightarrow C) = \left(\frac{w_1 * Supp(A \rightarrow C)}{minSupp} \right) \quad (7)$$

As in $Fitness_1$, $minSupp$ is also the minimum threshold value defined by the user. There are other fitness functions in the field [52, 55, 56], but they are essentially different combinations of support, confidence, and other measures of interest. A detailed description of various measures of interest usually applied in the association rule mining literature is available in [57].

The evaluation of each bacterium is related to the occurrence probability and accuracy of an association rule in the database. The selection of bacteria is proportional to their fitness values. The fitness function used in BaCARO-II and in the benchmark algorithms is

$$Fitness(A \rightarrow C) = w_1 * Supp(A \rightarrow C) + w_2 * Conf(A \rightarrow C) \quad (8)$$

where $w_1 = w_2 = 0.5$ and $w_1 + w_2 = 1$, subject to

$$A \cap C \neq \emptyset \quad \text{and} \quad |A| > 0, |A| > |C| \quad (9)$$

where $|\cdot|$ returns the cardinality of a set.

The algorithms use support and confidence to calculate the fitness value and the interestingness measure to compare them from a different perspective, as in [14, 58].

7. Experimental Results

To assess the performance of the algorithms, we run several experiments over distinct scenarios. The first set of tests was performed using five different binary static datasets and the second was run applying a sliding-window approach in the datasets to simulate the data streams. Finally, some experiments were performed investigating the computational complexity of the algorithms using a standard and an optimized architecture.

FIGURE 6: Intracellular communication in association rule mining.

The following algorithms were implemented for comparison: BFO; BaCARO-II; GA; and CLONALG, as well as their stream versions sBFO; sBaCARO-II; sGA; and sCLONALG [59]. All algorithms were implemented in Java 1.7.0.95 over a GNU/Linux environment (Debian 3.16.7-ckt20-1). The experiments were run in an Intel Pentium® Dual-core CPU t4500 @ 2.30GHz.

7.1. Performance Tests in Static Datasets. The BFO parameters were set as follows: $S = 100$, $N_{ed} = 10$, $N_{re} = 10$, $N_c = 10$, $N_s = 5$, and $P_{ed} = 0.4$. The BaCARO-II parameters were set as follows: $ic = 0.01$, $P_{cha} = 0.9$, $ec = 0.6$, and $Size_{net} = 2$. For CLONALG we used $N_1 = 20$, $N_2 = 5$, and $N_c = 10$, and, finally, for GA we used $P_{Crossover} = 0.6$ and $P_{Mutation} = 0.01$. All populations were set with 100 individuals and the maximum number of iterations was 100.

The following datasets were taken from the UCI Machine Learning Repository [60]: *SPECT Heart* database, with a sparsity of 66.75%; *Mushroom Database*, with 119 items and 8,124 instances with a sparsity of 80.67%; *Balance Scale Database*, with 23 items and 625 instances with a sparsity of 78.26%; *Flare Data*, with 49 items and 1,389 instances with a sparsity of 73.47%; and the *Monks Problems-1 Database*, with 19 items and 432 instances with a sparsity of 63.16%; and the

Nursery Database with 32 items and 12,960 instances with a measure of sparsity around 71.88%.

All the values taken over ten simulations of BFO, BaCARO-II, CLONALG, and GA for static environments are summarized in Table 1, while sBFO, sBaCARO-II, sCLONALG, and sGA for static and dynamic environments are summarized in Table 2. The values presented are the mean $\pm$ standard deviation and minimum and maximum values for the set of rules found in the final population of each algorithm over ten simulations, where S means support, C confidence, I interestingness, U number of unique rules found over the last set of candidate solutions, and $Time$ the processing time. As S and C are used in the fitness function, we selected the best fitness value from the final population. On the other hand, I is conceptually different from S and C and we used it to estimate the heterogeneity of solutions in the final population, as well as U .

In general, BaCARO-II presented better results than BFO, CLONALG, and GA in most measures. For instance, BaCARO-II overcomes BFO in all five datasets for the S and P measures. It occurs because BFO makes use of its global information by compounding a measure value of each attribute of the bacterium to influence the entire colony. BaCARO-II uses its global information to promote punctual

TABLE 1: Results for SPECT, Mushroom, Balance, Flare, Monks, and Nursery datasets were obtained by BFO, BaCARO-II, CLONALG, and GA. The values presented are the average $\pm$ standard deviation (minimal; maximal) taken over ten simulations.

	BFO	BaCARO-II	CLONALG	GA
SPECT				
S	0.40 $\pm$ 0.14(0.01;0.46)	0.43$\pm$0.04(0.36;0.46)	0.35 $\pm$ 0.09(0.16;0.46)	0.38 $\pm$ 0.07(0.28;0.46)
C	0.95 $\pm$ 0.01(0.92;1.00)	0.93 $\pm$ 0.01(0.907;0.94)	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)
I	0.43 $\pm$ 0.07(0.27;0.59)	0.41 $\pm$ 0.08(0.324;0.59)	0.60$\pm$0.08(0.49;0.74)	0.56 $\pm$ 0.15(0.46;0.99)
U	11.00 $\pm$ 4.76(1.00;17.00)	40 $\pm$ 2.40(36.00;43.00)	99.50 $\pm$ 0.97(97;100)	99.90$\pm$0.31(99;100)
P	18.50 $\pm$ 2.38(16.76;25.07)	3.11$\pm$0.16(2.967;3.56)	6.80 $\pm$ 0.42(6.27;7.50)	3.55 $\pm$ 0.10(3.46;3.73)
Mushroom				
S	0.00 $\pm$ 0.00(0.00;0.00)	0.01$\pm$0.01(0.00;0.03)	0.00 $\pm$ 0.00(0.00;0.0)	0.00 $\pm$ 0(0.00; 0.00)
C	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)	0.65 $\pm$ 0.24(0.50;1.00)	0.70 $\pm$ 0.25(0.50;1.00)
I	0.02 $\pm$ 0.03(0.00;0.12)	0.13$\pm$0.14(0.01;0.49)	0.05 $\pm$ 0.04(0.01;0.12)	0.05 $\pm$ 0.04(0.02;0.12)
U	95.50 $\pm$ 1.84(93.00;99.00)	15.70 $\pm$ 3.86(10.00;20.00)	100$\pm$0.00(100;100)	49.90 $\pm$ 0.87(48;51)
P	2577 $\pm$ 58.84(2451;2653)	535.7$\pm$13.81(503.1;557.6)	1196.9 $\pm$ 24.46(1166;1237)	644.08 $\pm$ 16.20(622;670)
Balance				
S	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)	0.02$\pm$0.01(0.00;0.04)	0.01 $\pm$ 0.01(0.00;0.04)
C	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)	0.37 $\pm$ 0.28(0.20;1.00)
I	0.02 $\pm$ 0.00(0.01;0.03)	0.05$\pm$0.01(0.01;0.07)	0.04 $\pm$ 0.03(0.01;0.14)	0.01 $\pm$ 0.01(0.00;0.039)
U	92.70 $\pm$ 2.83(88.00;96.00)	7.60 $\pm$ 1.71(5.00;10.00)	98.8$\pm$1.68(95;100)	4.40 $\pm$ 2.75(1;10)
P	29.84 $\pm$ 1.68(28.45;33.30)	5.73$\pm$0.14(5.50;5.99)	12.17 $\pm$ 1.26(11.50;15.63)	6.69 $\pm$ 0.19(6.41;6.97)
Flare				
S	0.06 $\pm$ 0.00(0.06;0.06)	0.70$\pm$0.33(0.15;0.96)	0.06 $\pm$ 0.00(0.06;0.06)	0.06 $\pm$ 0.00(0.06;0.06)
C	0.99 $\pm$ 0.00(0.97;1.00)	0.98 $\pm$ 0.02(0.94;1.00)	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)
I	0.253 $\pm$ 0.17(0.07;0.45)	0.28 $\pm$ 0.19(0.16;0.84)	0.68$\pm$0.27(0.43;0.99)	0.45 $\pm$ 0.02(0.40;0.49)
U	95.40 $\pm$ 2.06(91.00;98.00)	22.20 $\pm$ 6.54(12.00;33.00)	99.80$\pm$0.42(99;100)	50.50 $\pm$ 1.58(50;55)
P	151.9 $\pm$ 7.249(141.3;163.9)	28.72$\pm$1.05(26.38;29.96)	63.26 $\pm$ 4.01(58.75;69.51)	35.83 $\pm$ 1.87(33.18;39.31)
Monks				
S	0.00 $\pm$ 0.00(0.00;0.00)	0.08 $\pm$ 0.09(0.02;0.25)	0.14$\pm$0.03(0.08;0.16)	0.11 $\pm$ 0.04(0.04;0.16)
C	0.90 $\pm$ 0.21(0.50;1.00)	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)	0.97 $\pm$ 0.07(0.75;1.00)
I	0.00 $\pm$ 0.00(0.00;0.00)	0.17$\pm$0.11(0.04;0.37)	0.14 $\pm$ 0.03(0.07;0.21)	0.10 $\pm$ 0.02(0.07;0.13)
U	94.40 $\pm$ 2.06(92.00;98.00)	10.70 $\pm$ 2.58(8.00;15.00)	100$\pm$0.00(100;100)	50.4 $\pm$ 2.7568(43;53)
P	18.37 $\pm$ 0.57(17.45;19.37)	3.53$\pm$0.11(3.29;3.64)	7.54 $\pm$ 0.42(7.07;8.26)	3.88 $\pm$ 0.11(3.74;4.07)
Nursery				
S	0.00 $\pm$ 0.00(0.00;0.00)	0.17$\pm$0.17(0.00;0.33)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0(0.00;0.00)
C	0.46 $\pm$ 0.48(0.00;1.00)	1.00$\pm$0.00(1.00;1.00)	0.57 $\pm$ 0.25(0.33;1.00)	0.510 $\pm$ 0.288(0.300;1.00)
I	0.00 $\pm$ 0.00(0.00;0.00)	0.35$\pm$0.32(0.01;0.66)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)
U	95.70 $\pm$ 1.702(92.00;98.00)	15.20 $\pm$ 4.541(8.00;24.00)	99.70$\pm$0.48(99;100)	49.30 $\pm$ 1.05(47;51)
P	889.7 $\pm$ 34.70(846.1;950.3)	169.3$\pm$4.73(161.7;175.2)	401.3 $\pm$ 30.90(349.2;440.42)	207.4 $\pm$ 12.38(184.9;222.7)

variations along the colony and improve its search ability. By improving it, BaCARO-II tends to maintain many agents over the same high adaptable regions. Consequently, BFO sometimes overcomes BaCARO-II in the U measure by applying more local search steps, avoiding the concentration of large numbers of agents in the same region. On the other hand, BFO makes less use of global information and then BaCARO-II presents better fitness values as well as processing time.

BaCARO-II presented competitive results for all datasets. The best performance of our bacterial algorithm was for the *Mushroom*, *Monks*, and *Nursery* databases. The average values of support, confidence, and interestingness of our

approach are higher than those presented by BFO. However, the number of rules generated by BaCARO-II is not greater than that of BFO in most datasets. On the other hand, our approach produces association rules with higher values of support and confidence. Another favourable point for BaCARO-II is its average processing time, which is smaller than its competitors. Nevertheless, BaCARO-II performs worse than BFO, GA, and CLONALG for all databases for the unique rules measure.

7.2. Bacterial Colony Algorithms in Stream Data. The same parameter configurations adopted in the static environment

TABLE 2: Results for *streamSPECT*, *streamMushroom*, *streamBalance*, *streamFlare*, *streamMonks*, and *streamNursery* datasets were obtained by sBFO, sBaCARO-II, sCLONALG, and sGA. The values presented are the average $\pm$ standard deviation (minimal; maximal) taken over ten simulations.

	sBFO	sBaCARO-II	sCLONALG	sGA
<i>streamSPECT</i>				
S	0.61$\pm$0.00(0.61;0.62)	0.41 $\pm$ 0.05(0.34;0.46)	0.36 $\pm$ 0.00(0.36;0.36)	0.351 $\pm$ 0.016(0.310;0.360)
C	1.00$\pm$0.00(1.00;1.00)	0.94 $\pm$ 0.03(0.87;0.97)	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)
I	0.63 $\pm$ 0.07(0.54;0.72)	0.82 $\pm$ 0.21(0.49;0.99)	0.82 $\pm$ 0.13(0.72;0.98)	0.963$\pm$0.083(0.727;0.99)
U	100$\pm$0.00(100;100)	41.30 $\pm$ 4.44(36.00;48.00)	99.90 $\pm$ 0.31(99;100)	99.70 $\pm$ 0.483(99.00;100.0)
P	100.9 $\pm$ 1.85(97;104)	2.70$\pm$0.10(2.63;2.90)	5.36 $\pm$ 0.30(4.99;5.83)	3.200 $\pm$ 0.077(3.051;3.340)
<i>streamMushroom</i>				
S	0.01$\pm$0.00(0.01;0.01)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)
C	1.00$\pm$0.00(1.00;1.00)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)
I	0.99$\pm$0.00(0.99;0.99)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)
U	100$\pm$0.00(100;100)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)	0.00 $\pm$ 0.00(0.00;0.00)
P	7662.3$\pm$290.5(7089;7994)	866.5 $\pm$ 65.234(792.8;990.2)	1386.5 $\pm$ 70.930(1290.2;1497.2)	872.0 $\pm$ 43.39(796.1;927.2)
<i>streamBalance</i>				
S	0.01 $\pm$ 0.00(0.01;0.01)	0.09$\pm$0.08(0.01;0.20)	0.03 $\pm$ 0.03(0.00;0.130)	0.02 $\pm$ 0.02(0.00;0.05)
C	1.00$\pm$0.00(1.00;1.00)	0.73$\pm$0.34(0.200;1.00)	0.27 $\pm$ 0.27(0.000;1.00)	0.216 $\pm$ 0.29(0.00;1.00)
I	0.13 $\pm$ 0.06(0.04;0.19)	0.14$\pm$0.06(0.04;0.20)	0.03 $\pm$ 0.04(0.000;0.13)	0.02 $\pm$ 0.02(0.00;0.04)
U	95.8$\pm$2.898(92;100)	7.30$\pm$3.02(1.00;11.00)	2.60 $\pm$ 2.06(0.00;6.00)	1.80 $\pm$ 2.25(0.00;7.00)
P	117.2 $\pm$ 5.391(111;126)	6.51 $\pm$ 0.40(6.05;7.24)	11.34 $\pm$ 0.43(10.64;12.01)	6.35$\pm$0.17(6.15;6.70)
<i>streamFlare</i>				
S	0.160 $\pm$ 0.000(0.160;0.160)	0.96$\pm$0.02(0.92;0.98)	0.02 $\pm$ 0.00(0.02;0.03)	0.02 $\pm$ 0.00(0.02;0.03)
C	1.00$\pm$0.00(1.00;1.00)	0.99 $\pm$ 0.02(0.93;1.00)	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)
I	0.990$\pm$0.000(0.990;0.990)	0.21 $\pm$ 0.027(0.17;0.25)	0.31 $\pm$ 0.02(0.28;0.32)	0.32 $\pm$ 0.02(0.26;0.32)
U	100 $\pm$ 0.00(100;100)	17.60 $\pm$ 2.633(13.00;22.00)	21.10 $\pm$ 4.28(15.00;30.00)	13.60 $\pm$ 3.71(9.00;20.00)
P	557.60 $\pm$ 20.76(532;607)	33.45$\pm$0.70(32.21;34.06)	66.79 $\pm$ 2.926(15;30)	36.72 $\pm$ 1.43(33.43;38.12)
<i>streamMonks</i>				
S	0.01 $\pm$ 0.00(0.01;0.01)	0.42$\pm$0.14(0.24;0.60)	0.28 $\pm$ 0.09(0.15;0.48)	0.21 $\pm$ 0.12(0.020;0.480)
C	1.00$\pm$0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)	0.98 $\pm$ 0.06(0.80;1.00)	0.96 $\pm$ 0.12(0.60;1.00)
I	0.07 $\pm$ 0.02(0.04;0.09)	0.34 $\pm$ 0.08(0.24;0.44)	0.24 $\pm$ 0.05(0.17;0.30)	0.20 $\pm$ 0.062(0.12;0.30)
U	97.70$\pm$1.33(96.00;100)	12.10 $\pm$ 3.17(7.00;17.00)	76.40 $\pm$ 6.80(65.00;88.00)	47.00 $\pm$ 3.55(39.00;50.00)
P	65.50 $\pm$ 1.77(63.00;69.00)	3.97 $\pm$ 0.14(3.68;4.15)	6.89 $\pm$ 0.14(6.64;7.08)	3.79$\pm$0.05(3.70;3.88)
<i>streamNursery</i>				
S	0.01 $\pm$ 0.00(0.01;0.01)	0.93$\pm$0.20(0.34;1.00)	0.01 $\pm$ 0.02(0.00;0.06)	0.02 $\pm$ 0.05(0.00;0.18)
C	1.00 $\pm$ 0.00(1.00;1.00)	1.00$\pm$0.00(1.00;1.00)	0.10 $\pm$ 0.13(0.00;0.33)	0.10 $\pm$ 0.18(0.00;0.52)
I	0.21 $\pm$ 0.22(0.03;0.61)	0.28$\pm$0.13(0.24;0.66)	0.01 $\pm$ 0.02(0.00;0.06)	0.05 $\pm$ 0.13(0.00;0.43)
U	1.00 $\pm$ 0.00(1.00;1.00)	11.80$\pm$3.583(4.00;18.00)	0.50 $\pm$ 0.70(0.00;2.00)	0.50 $\pm$ 0.70(0.00;2.00)
P	9740.4 $\pm$ 480.1(9069;10312)	409.2$\pm$11.09(394.7;432.2)	633.3 $\pm$ 35.94(569.2;686.8)	458.7 $\pm$ 30.89(383.9;481.6)

were applied to the dynamical case. As datasets have different sizes, we fixed the sliding-window size at 100, changing 1 object per iteration.

By considering the highlighted performance of our algorithm presented here and in other works [5, 6], we designed dynamical environments to evaluate its robustness and flexibility in mining association rules. In fact, we converted the following static datasets, SPECT, Balance Scale, Flare, Monks, and Nursery, to dynamical datasets by applying the *Sliding-Window* approach over them. To differentiate static and stream databases, we refer to the stream versions

as *streamSPECT*, *streamBalance*, *streamFlare*, *streamMonks*, and *streamNursery*.

For experimental proposes, we fixed the sliding-window size at 100 objects per time step t_i of the data stream and its transition from t_i to t_{i+1} occurs when one object from the stream enters and another leaves the sliding-window, which always maintains its size. The sliding-window schema, data stream, and its transactions used in the experiments are illustrated in Figure 7.

The results obtained by the stream versions of the algorithms (sBaCARO-II, sCLONALG, sBFO, and sGA) in

FIGURE 7: Sliding-window approach for association rule mining.

the dynamic environments for *streamSPECT*, *streamBalance*, *streamFlare*, *streamMonks*, and *streamNursery* output are summarized in Table 2.

Although the final result is based on the different objects that run through the sliding-window during the association rule mining process, it is undeniable that the objects at the final time t are the most relevant for the development of the previous ones.

To validate the results obtained in static and dynamic environments we compared the results of our approach with BFO; we choose this one instead of GA or CLONALG due to its superior performance during experimental results, using Student's t-test with two-tailed distribution. In the static environment, for the Balance database, the t-test showed no statistical difference for the highest values of the support and confidence measures, $8.53e^6$ and 0, respectively; the t-test for the Flare database indicates, respectively, the statistical difference of 0.00017 and 0.1341 for the measures of support and confidence; for the Monks database, the value obtained by the statistical difference t-test for the support was 0.015, while for the confidence it was 0.167; already in the Mushroom database, the t-test registered 0.025 for the support measure, while it did not record a difference for the confidence measure; for the Nursery database, the values indicated by the t-test were 0.010 and 0.006 for the confidence and support measures, respectively; and finally, for the SPECT database, the t-test pointed to the largest statistical difference between the algorithms, the support recorded with 0.489 for the support measure and 0.109 for the confidence measure.

In the dynamic environment, the t-test for the Balance database registered 0.009 and 0.041 for the support and confidence measures, respectively; for the Flare database, values of $2.2e^{15}$ and 0.343 were, respectively, recorded for support and confidence; already in the Monks database, the t-test for the support was pointed out with $7.93e^6$ and showed no statistical difference for the confidence measure; for the Mushroom database the t-test showed statistical difference for both measures because sBaCARO-II did not generate

any rule; and finally, the t-test indicated $3.73e^7$ and 0.0006, respectively, for support and confidence measures.

7.3. Some Notes on BaCARO-II Running Time. To assess the running time of the proposed algorithm, we tested its static version using a different hardware and software architecture: an accelerating performance for server-side Java [61] applications, an optimization on JVM (Java Virtual Machine) from version 1.8 to newer versions to Intel® new Xeon Scalable Processors. We performed new experiments aiming at investigating Intel's High Performance Computing (HPC) platforms benefits. These new experiments were made on a compute node composed of two Intel® Xeon® Platinum 8160 processors @ 2.10 GHz, each one with 24 physical cores (48 logical) and 33 MB of cache memory, 190 GB of RAM, two Intel® Solid State Drive Data Center (Intel® SSD DC) S3520 SERIES with 1.2 TB e 240 GB store capacity, and a CentOS 7 operation system running kernel version 3.10.0-693.21.1.317.x86_64. Table 3 provides a comparison of the running times of BaCARO-II for the static datasets in both architectures. As can be observed, the use of an HPC platform leads to an average 2.60-fold gain in performance.

8. Final Remarks and Future Trends

There are many phenomena happening in a bacterial colony. Some of them, such as foraging and chemotaxis, were used to construct tools to solve complex problems. This paper proposed and applied a new bacteria-inspired algorithm by looking at intra- and extracellular communication networks, as well as interactions between bacteria and their internal constituent parts to deal with association rule mining. The results presented by BaCARO-II showed a superior performance to other bio-inspired algorithms, such as BFO, GA, and CLONALG when applied to the same tasks.

With the current need of solving stream data problems, we designed and applied versions of BFO, BaCARO-II, GA, and CLONALG for mining association rules in stream data. The proposed bacterial approach showed good results in the

TABLE 3: Running time comparison of different architectures.

Dataset	Intel® Pentium®	Intel® Xeon® Platinum
SPECT	3.110±0.167 (2.967; 3.561)	1.10±0.02 (1.05; 1.13)
Mushroom	535.7±13.81 (503.1; 557.64)	174.47±2.95 (170.79; 180.82)
Balance	5.731±0.146 (5.507; 5.995)	2.26±0.05 (2.2; 2.4)
Flare	28.72±1.056 (26.38; 29.96)	9.82±0.13 (9.6; 10.01)
Monks	3.533±0.119 (3.293; 3.643)	1.44±0.05 (1.39; 1.52)
Nursery	169.3±4.735 (161.7; 175.2)	92.76±1.62 (90.14; 95.42)

experiments performed, in both static and stream data. We understand that the superior performance of our approach is primarily due to two reasons: first, the local search performed in the intracellular communication phase and, second, the use of information available in the neighbourhood (nearest bacterial cell) of each bacterial cell to improve the search space exploration. BFO was very competitive and presented better results in some dynamic scenarios, though it demands longer processing time.

As future investigations, sBaCARO-II should be applied to stream data mining tasks with different kinds of stream data processing models, *Landmark* and *Damped*. Other settings for the *Sliding-Windows* size should also be tested and the results compared with other algorithms, such as the ones presented in [62, 63]. Future works may also include a deeper understanding of bacterial behaviours and phenomena.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank CAPES, CNPq, Fapesp, and Mackpesquisa for the financial support. The authors also acknowledge the support of Intel for the Natural Computing and Machine Learning Laboratory as an Intel Center of Excellence in Artificial Intelligence.

References

- [1] M. Matsushita and H. Fujikawa, "Diffusion-limited growth in bacterial colony formation," *Physica A: Statistical Mechanics and its Applications*, vol. 168, no. 1, pp. 498–506, 1990.
- [2] J. Van Helden, A. Toussaint, and D. Thieffry, "Bacterial molecular networks: Bridging the gap between functional genomics and dynamical modelling," *Methods in Molecular Biology*, vol. 804, pp. 1–11, 2012.
- [3] E. Ben-Jacob, "Learning from bacteria about natural information processing," *Annals of the New York Academy of Sciences*, vol. 1178, pp. 78–90, 2009.
- [4] R. S. Xavier, N. Omar, and L. N. De Castro, "Bacterial colony: Information processing and computational behavior," in *Proceedings of the 2011 3rd World Congress on Nature and Biologically Inspired Computing, NaBIC 2011*, pp. 439–443, Spain, October 2011.
- [5] D. S. da Cunha, R. S. Xavier, and L. N. de Castro, "A bacterial colony algorithm for association rule mining," in *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'15)*, 2015.
- [6] D. S. Da Cunha, R. S. Xavier, D. G. Ferrari, and L. N. De Castro, "Association rule mining using a bacterial colony algorithm," in *Proceedings of the 2nd Latin-America Congress on Computational Intelligence, LA-CCI 2015*, Brazil, October 2015.
- [7] D. S. da Cunha and L. N. de Castro, "Evolutionary and immune algorithms applied to association rule mining," in *Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO)*, Bhubaneswar, 2012.
- [8] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.
- [9] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, 1992.
- [10] L. N. de Castro and F. J. von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.
- [11] D. S. da Cunha, R. S. Xavier, D. G. Ferrari, and L. N. de Castro, "Bacterial Colony Algorithms Applied to Association Rule Mining in Static Data and Streams," in *Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems*, pp. 525–533, Springer, 2018.
- [12] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*, pp. 207–216, May 1993.
- [13] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference Very Large Data Bases (VLDB'94)*, 1994.
- [14] S. Dehur, A. K. Jagadev, A. Ghosh, and R. Mall, "Multi-objective genetic algorithm for association rule mining using a homogeneous dedicated cluster of workstations," *American Journal of Applied Sciences*, vol. 3, no. 11, pp. 2086–2095, 2006.
- [15] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan, *Data Mining: A Knowledge Discovery Approach*, Springer Science and Business Media, 2007.
- [16] N. Jiang and L. Gruenwald, "Research issues in data stream association rule mining," *ACM SIGMOD Record*, vol. 35, no. 1, pp. 14–19, 2006.
- [17] C. C. Aggarwal, *Data Streams: Models and Algorithms*, vol. 31, Springer Science and Business Media, 2007.

- [18] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 18–26, 2005.
- [19] S. Guha, N. Koudas, and K. Shim, "Data-streams and histograms," in *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pp. 471–475, Herssonissos, Greece, 2001.
- [20] Y. Zhu and D. Shasha, "Statstream: Statistical monitoring of thousands of data streams in real time," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, Hong Kong, 2002.
- [21] M. H. Le Gruenwald, "Estimating missing values in related sensor data streams," in *COMAD*, 2005.
- [22] E. D. Demaine, A. López-Ortiz, and J. I. Munro, "Frequency estimation of internet packet streams with limited space," in *Proceedings of the European Symposium on Algorithms*, 2002.
- [23] Y. D. Cai, D. Clutter, G. Pape, J. Han, M. Welge, and L. Auvil, "MAIDS: Mining alarming incidents from data streams," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD 2004*, pp. 919–920, France, June 2004.
- [24] D. Lee and W. Lee, "Finding maximal frequent itemsets over online data streams adaptively," in *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 266–273, Houston, TX, USA, 2005.
- [25] H. Huang, X. Wu, and R. Relue, "Association analysis with one scan of databases," in *Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002*, pp. 629–632, Maebashi City, Japan, 2002.
- [26] R. Relue, X. Wu, and H. Huang, "Efficient runtime generation of association rules," in *Proceedings of the tenth International Conference on Information and Knowledge Management (IKM'01)*, p. 466, Atlanta, Georgia, USA, October 2001.
- [27] L. Yang and M. Sanver, "Mining short association rules with one database scan," in *Proceedings of the International Conference on Information and Knowledge Engineering, IKE'04*, pp. 392–395, USA, June 2004.
- [28] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 226–235, Washington, DC, USA, August 2003.
- [29] I. Habibi, E. S. Emamian, and A. Abdi, "Quantitative analysis of intracellular communication and signaling errors in signaling networks," *BMC Systems Biology*, vol. 8, no. 1, 2014.
- [30] J. F. Prescott and P. M. Dowling, *Antimicrobial Therapy in Veterinary Medicine*, John Wiley & Sons, 2013.
- [31] B. Alberts, *Molecular Biology of the Cell*, CRC Press, 2017.
- [32] E. Ben Jacob, Y. Shapira, and A. I. Tauber, "Seeking the foundations of cognition in bacteria: From Schrödinger's negative entropy to latent information," *Physica A: Statistical Mechanics and its Applications*, vol. 359, no. 1–4, pp. 495–524, 2006.
- [33] H. Salis, A. Tamsir, and C. Voigt, "Engineering bacterial signals and sensors," in *Bacterial Sensing and Signaling*, vol. 16, pp. 194–225, Karger Publishers, 2009.
- [34] H. J. Bremermann, "Chemotaxis and optimization," *Journal of The Franklin Institute*, vol. 297, no. 5, pp. 397–404, 1974.
- [35] B. Niu and H. Wang, "Bacterial colony optimization: principles and foundations," in *Proceedings of the International Conference on Intelligent Computing*, vol. 304, 2012.
- [36] K. M. Passino, "Bacterial foraging optimization," in *Innovations and Developments of Swarm Intelligence Applications*, vol. 219, IGI Global, 2012.
- [37] B. Xing and W. Gao, "Bacteria inspired algorithms," in *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms*, vol. 62, pp. 21–38, Springer, 2014.
- [38] S. Das, A. Biswas, S. Dasgupta, and A. Abraham, "Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications," in *Foundations of Computational Intelligence*, vol. 203, pp. 23–25, Springer, 2009.
- [39] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "A synergy of differential evolution and bacterial foraging optimization for global optimization," *Neural Network World*, vol. 17, no. 6, pp. 607–626, 2007.
- [40] E. A. H.-O. B. Mezura-Montes, "Modified bacterial foraging optimization for engineering design," in *Intelligent Engineering Systems through Artificial Neural Networks*, pp. 1–8, ASME Press, 2009.
- [41] S. M. Abd-Elazim and E. S. Ali, "A hybrid particle swarm optimization and bacterial foraging for optimal power system stabilizers design," *International Journal of Electrical Power & Energy Systems*, vol. 46, no. 1, pp. 334–341, 2013.
- [42] S. M. Abd-Elazim and E. S. Ali, "Bacteria foraging optimization algorithm based svc damping controller design for power system stability enhancement," *International Journal of Electrical Power & Energy Systems*, vol. 43, no. 1, pp. 933–940, 2012.
- [43] K. S. Kumar and T. Jayabarathi, "Power system reconfiguration and loss minimization for an distribution systems using bacterial foraging optimization algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 36, no. 1, pp. 13–17, 2012.
- [44] S. M. Abd-Elazim and E. S. Ali, "Synergy of particle swarm optimization and bacterial foraging for TCSC damping controller design," *International Journal of World Scientific and Engineering Academy and Society (WSEAS) Transactions on Power Systems*, vol. 8, pp. 74–84, 2013.
- [45] H. Chen, Y. Zhu, and K. Hu, "Multi-colony bacteria foraging optimization with cell-to-cell communication for RFID network planning," *Applied Soft Computing*, vol. 10, no. 2, pp. 539–547, 2010.
- [46] M. Wan, L. Li, J. Xiao, C. Wang, and Y. Yang, "Data clustering using bacterial foraging optimization," *Journal of Intelligent Information Systems*, vol. 38, no. 2, pp. 321–341, 2012.
- [47] J. R. Olesen, J. Cordero, and Y. Zeng, "Auto-clustering using particle swarm optimization and bacterial foraging," in *Proceedings of the International Workshop on Agents and Data Mining Interaction*, pp. 69–83, 2000.
- [48] R. Majhi, G. Panda, B. Majhi, and G. Sahoo, "Efficient prediction of stock market indices using adaptive bacterial foraging optimization (ABFO) and BFO based techniques," *Expert Systems with Applications*, vol. 36, no. 6, pp. 10097–10104, 2009.
- [49] S. R. Chhabra, B. Philipp, L. Eberl, M. Givskov, P. Williams, and M. Cámara, "Extracellular communication in bacteria," in *The Chemistry of Pheromones and Other Semiochemicals II*, pp. 279–315, Springer, Berlin, Heidelberg, Germany, 2005.
- [50] C. G. Bowsher and P. S. Swain, "Environmental sensing, information transfer, and cellular decision-making," *Current Opinion in Biotechnology*, vol. 28, pp. 149–155, 2014.
- [51] T. Bäck, D. Fogel, and Z. Michalewicz, *Evolutionary Computation I: Basic Algorithms and Operators*, vol. 1, CRC Press, 2000.
- [52] Y. Su, X. Gu, and Z. Li, "Incremental updating algorithm based on artificial immune system for mining association rules," in

- Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'06)*, Shanghai, China, October 2006.
- [53] Y. Zhang, S. Bu, and Y. Zhang, "Association rules mining based on the improved immune algorithm," in *Proceedings of the Third International Symposium on Intelligent Information Technology Application*, 2009.
 - [54] Y. Zhang and S. Bu, "Association rules mining based on simulated annealing immune programming algorithm," in *Proceedings of the International Conference on Computer Engineering and Technology*, 2009.
 - [55] T. Liu, "An immune based association rule algorithm," in *Proceedings of the Second International Conference on Innovative Computing, Information and Control (ICICIC 2007)*, Kumamoto, Japan, 2007.
 - [56] Z. Lei and L. Ren-hou, "An algorithm for mining fuzzy association rules based on immune principles," in *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering*, 2007.
 - [57] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: a survey," *ACM Computing Surveys*, vol. 38, no. 3, pp. 1–32, 2006.
 - [58] M. J. del Jesus, J. A. Gámez, P. González, and J. M. Puerta, "On the discovery of association rules by means of evolutionary algorithms," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 5, pp. 397–415, 2011.
 - [59] D. S. da Cunha and L. N. de Castro, "Evolutionary and immune algorithms applied to association rule mining in static and stream data," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Brazil, 2018.
 - [60] K. Bache and M. Lichman, *UCI - Machine Learning Repository*, 2013, <http://archive.ics.uci.edu/ml>.
 - [61] I. Corporation, "Accelerating performance for server-side Java applications," Portland, 2017.
 - [62] P. Deepa Shenoy, K. G. Srinivasa, K. R. Venugopal, and L. M. Patnaik, "Evolutionary approach for mining association rules on dynamic databases," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2003.
 - [63] K. R. Venugopal, K. G. Srinivasa, and L. M. Patnaik, "Dynamic association rule mining using genetic algorithms," in *Soft Computing for Data Mining Applications*, 2009.