# Evolutionary computation and swarm intelligence for the inference of gene regulatory networks

Leandro Takeshi Hattori and Heitor Silvério Lopes*

Graduate Program in Electrical and Computer Engineering,
Federal University of Technology,
Paraná, Curitiba, Brazil
Email: leandrohattori@alunos.utfpr.edu.br
Email: hslopes@utfpr.edu.br
*Corresponding author

Fabrício Martins Lopes

Graduate Program in Bioinformatics,
Federal University of Technology,
Paraná, Cornélio Procópio, Brazil
Email: fabricio@utfpr.edu.br

**Abstract:** The inference of gene regulatory networks (GRNs) from expression profiles is still an important challenge in bioinformatics research. The main difficulty of this problem is associated to the huge number of genes and the small number of samples available, as well as the intrinsic noise in the data acquisition process. In this context, this paper presents a feature selection approach to the identification of GRNs using optimisation strategies from evolutionary computation and swarm intelligence. As a case-study we used an artificial gene network (AGN) based on the *scale-free* topology. This AGN has 1,000 genes and was simulated with 500 temporal expression samples. The methods compared were: differential evolution (DE), bat algorithm (BAT) and artificial bee colony (ABC) algorithms. All algorithms used their standard control parameters and the same criterion function: the mean conditional entropy (MCE). This is an information theory measure, commonly adopted for various feature selection problems in the pattern recognition research field. The results showed that DE algorithm leaded to the best results than BAT and ABC in all comparisons, and the inferred network was more similar to the original network.

**Keywords:** evolutionary computation; swarm intelligence; bioinformatics; gene regulatory networks; GRNs; complex networks.

**Reference** to this paper should be made as follows: Hattori, L.T., Lopes, H.S. and Lopes, F.M. (2016) 'Evolutionary computation and swarm intelligence for the inference of gene regulatory networks', *Int. J. Innovative Computing and Applications*, Vol. 7, No. 4, pp.225–235.

**Biographical notes:** Leandro Takeshi Hattori received his MSc in Computer Science and currently, he is a PhD student at the Graduate Program in Electrical Engineering and Computer Science of the Federal University of Technology Paraná, Curitiba, Brazil.

Heitor Silvério Lopes received his degrees in Electronic Engineering (1984) and MSc in Biomedical Engineering (1990) both from the Federal University of Technology – Paraná – UTFPR, and PhD in Electrical Engineering (1996) from the Federal University of Santa Catarina. He has published more than 200 refereed papers in conferences and journals and was the advisor for 50 MSc and PhD students. Currently he is a Titular (Full) Professor of the Department of Electronics, at UTFPR, campus Curitiba, Brazil, with appointments in Computer Engineering. He is also the Head of the Bioinformatics & Computational Intelligence Laboratory. His current research interests are: evolutionary computation, bioinformatics, computer vision, deep learning and data-mining.

Fabricio M. Lopes received his PhD in Bioinformatics from University of Sao Paulo (USP), Brazil in 2011. His PhD thesis received an honorable mention from USP Thesis Award 2011. He is Associate Professor at the Computer Science Department at Federal University of Technology – Paraná (UTFPR), Brazil. His main research topics are in systems biology and pattern recognition for bioinformatics and image processing, including but not limited to, gene network identification data integration and complex network analysis. He is an advisor of the Graduate Bioinformatics Program (PPGBIOINFO) and Graduate Program in Informatics (PPGI), both of UTFPR. His recent work can be accessed at http://pessoal.utfpr.edu.br/fabricio/.

This paper is a revised and expanded version of a paper entitled 'A discretized differential evolution algorithm for the inference of gene regulatory networks' presented at Latin America Congress on Computational Intelligence (LA-CCI), Curitiba, Brazil, 13–16 October 2015.

## 1 Introduction

The deoxyribonucleic acid (DNA), present in the cells of living organisms, carries essential genetic instructions for supporting life. Genes are portions of DNA that encode the 'recipe' for a given biological product, such as a protein. In simple words, when a gene expresses, it induces the production of a protein that have some important task in the organism. Such tasks can be related to transport, catalysis, signalling and structure maintenance, for instance. Also, proteins can be also responsible for regulating the expression of another genes. Therefore, genes products can be viewed as inhibitory or excitatory factors for the expression of other genes. When observed as a whole, organisms have a dense network of genes regulating other genes and this is known as gene regulatory networks (GRNs) (Shmulevich and Dougherty, 2007; Kelemen et al., 2008).

These networks are helpful for understanding the underlying complex biochemical mechanisms that take place within cells and, eventually, may lead to a better understanding of the behaviour of diseases, based on the behaviour of gene expressions (Chandra and Padiadpu, 2013).

Thanks to the rapid development of massive biological data collection techniques in the last years, such as DNA microarrays and RNA-Seq (Wang et al., 2009), it is possible to evaluate the expression of thousands of genes simultaneously for a given organism. On the other hand, the increase of the number of genes that can be observed leads to an exponential growth of the number of possible interactions (connections) between them, i.e., the GRNs. Therefore, the inference of GRNs from gene expression data is a real-world challenging problem, for which there is no closed solution for large-scale problems, from the computational point of view.

When observing the behaviour of a gene as a function of the expression of a (large) set of genes, a large number of possible combinations can be considered for a cause-effect relationship. This problem can be understood as a feature selection problem, such that a specific target gene can be inhibited or excited depending upon the joint effect of several other genes. So, the key issue is to find which genes, out of a large set, have direct influence in the target gene. Several methods for dealing with the feature selection problem were proposed, and some are very popular (Lopes et al., 2008, 2014a; Marbach et al., 2012; Jimenez et al., 2015). Usually, feature selection methods include two elements: a criterion function to evaluate a set of candidate genes and a search algorithm that browses the space of feasible combinations of genes. Taking into account the exponential complexity of the feature selection problem, metaheuristic methods lead to a better cost-benefit relationship (computational cost versus the solution quality) as the size of the problem increases.

Another important issue in the GRNs inference process is how to validate the methods. Computational methods that simulate GRNs from topologies defined from theoretical models of complex networks can be a solution for evaluating the inference methods, such as the artificial gene networks (AGNs) (Lopes et al., 2011a; Byrne et al., 2014). The AGNs can represent complex networks topologies such as small-world, scale-free and uniform-random. The gene expression data generated from AGNs are produced by means of a transition function based on probabilistic Boolean functions for the input signal (Shmulevich et al., 2002). The main advantage of using AGNs is a clearly defined structure that allows its comparison with the inferred network by the method under evaluation. Also, it is possible to generate AGNs with different features and structures, so as to allow the study of robustness of inference methods, by varying the number of genes in the network, the average number of connections between genes, and the topology of network. Following a previous work (Hattori et al., 2015), we also used AGNs for creating an artificial benchmark to test the algorithms. However, differently from that work, here, we focus on the comparison of metaheuristic methods for the inference of GRNs.

Therefore, the objective of this work is to analyse the performance of evolutionary computation (EC) and swarm intelligence (SI) methods for the inference of GRNs. Three algorithms are compared: differential evolution (DE) algorithm, artificial bee colony (ABC), and the bat algorithm (BAT). These algorithms have been used in many hard optimisation problems, in many areas, including bioinformatics (Alba et al., 2007; Das et al., 2008; Kalegari and Lopes, 2013; Parpinelli et al., 2014).

The remaining of this paper is organised as follows. Section 2 first describes GRNs and their inference process. Then, in Sections 3 and 4, a brief review of the EC and SI algorithms used to unveil the complex relationship among genes of a GRN are presented. Later, in Section 5, the specific methods used in this work are described in details. Section 6 reports the experiments done and the results obtained. Finally, in Section 7, conclusions drawn from the experiments are presented and future research directions are pointed out.

## 2 Gene regulatory networks

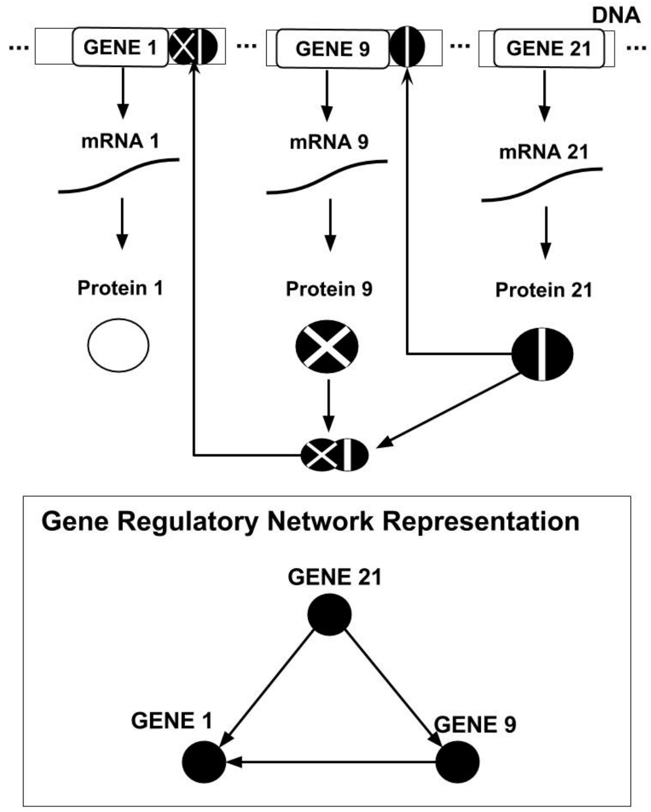
The regulation of biological functions in an organism takes place according to the GRNs, as shown in Figure 1. Such regulation takes place by means of the expression of a given gene or a set of genes and, the corresponding product (or combination with another gene products) will induce changes in the expression of another gene. The genes which expression affect the behaviour of another gene are called predictors, and those which are affected by them are called target genes. The next two sections focus, respectively, on the inference process of GRNs and how AGNs can be useful for the comprehension of biological functions.

## 2.1 *Inference of GRNs*

The reverse engineering process of GRNs consists in inferring a model of influence network among a set of genes.

Thanks to the advancement of molecular biology methods, and the consequent large availability of biological data, GRNs are becoming increasingly complex, aiming at representing with more details the regulatory system. Here, there is a great challenge for bioinformatics: how to infer complex GRNs from time series gene expression data, taking into account a huge number of genes and a very small number of samples. Some methods were developed for this purpose, and the reader is directed to see, for instance, Terfve et al. (2012) and Lopes et al. (2009, 2011a, 2011b).

**Figure 1** Representation of the biological model of genic regulation between genes 1, 9 and 21



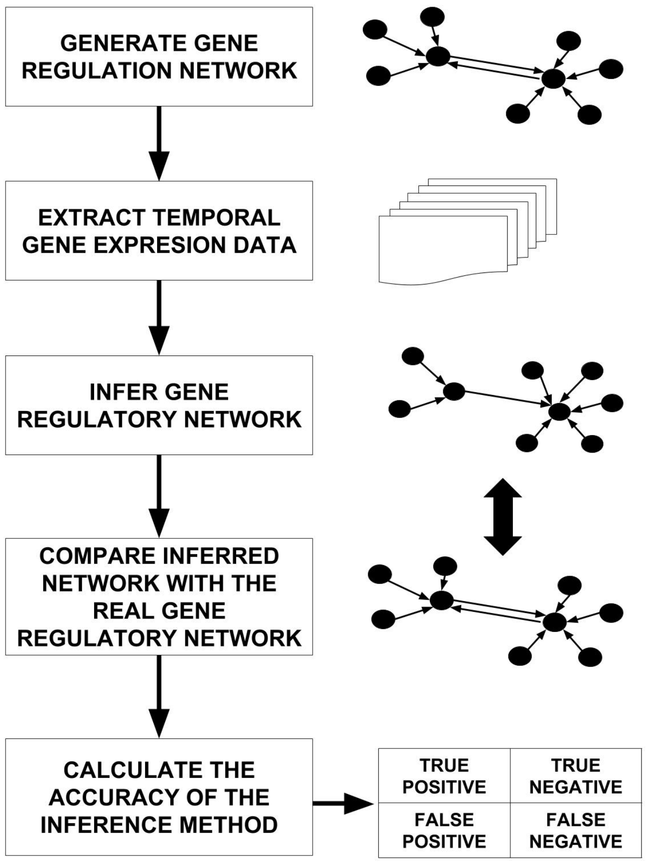
Notes: It is possible to notice that gene 1 is the target because it is influenced by the expression of other genes (predictors), in this case, genes 21 and 9. Also, gene 21 is a predictor of gene 9, as well. So, a complex network of genes regulating other genes emerge and it is represented by the directed graph right below.

An objective way to test the GRN inference method is the reconstruction of the network using artificially generated temporal data (gene-expression data), starting from a known GRN (see Section 2.2). This procedure is used in this work and it is depicted in Figure 2.

A key issue in the inference of GRNs is how to evaluate genes that are candidate predictors to target genes. For this purpose, some criterion functions were proposed in the literature, such as: statistical correlations, mean conditional entropy (MCE), mutual information, and combinations of biological scores (Barrera et al., 2007; Lopes et al., 2009, 2014b; Marbach et al., 2012).

Differently from correlation measures, the MCE can evaluate *N* candidate predictor genes simultaneously, instead of one at a time and, so, it evaluates the quality of a whole set. Since the criterion function evaluates a subset of the predictor genes, an efficient search algorithm is necessary for browsing the space of all possible combinations of genes. Here, we used EC and SI methods, explained later, together with the MCE, which was already reported to be efficient with EC algorithms (Jimenez et al., 2015).

**Figure 2** The inference procedure using GRN


GENERATE GENE REGULATION NETWORK
EXTRACT TEMPORAL GENE EXPRESION DATA
INFER GENE REGULATORY NETWORK
COMPARE INFERRED NETWORK WITH THE REAL GENE REGULATORY NETWORK
CALCULATE THE ACCURACY OF THE INFERENCE METHOD
TRUE POSITIVE
TRUE NEGATIVE
FALSE POSITIVE
FALSE NEGATIVE

## 2.2 *Artificial gene networks*

Although several methods have been proposed in the literature for modelling and identifying GRNs from expression profiles, usually, the information necessary to validate the inferred networks is incomplete or unknown.

More specifically, when considering biological networks, they are static and it is not possible to test variations about them in a suitable way, such as to test different network topologies, number of genes, average number of connections per gene, number of expression profile available, to cite but a few.

To circumvent this problem, in this work we used an objective approach to generate AGNs, as proposed by Lopes et al. (2011a). The AGNs have topologies defined by theoretical models of complex networks, such as: uniformly-random networks proposed by Erdös and Rényi (1959), small-world networks proposed by Watts and Strogatz (1998) and scale-free networks proposed by Barabási and Albert (1999). A comparison of these topologies can be found in Hattori et al. (2015), where the reader can find a deeper discussion about them.

A GRN is modelled as a graph, where genes represent the set of nodes *v*, and the set of edges *k* represents the connections among them. In this work, the size *n* of real networks represents the number of genes, and *k* represents the average number of connections (edges) for each gene.

It is of particular interest the scale-free model (Barabási-Albert – BA), which is frequently found in biological networks (Stuart et al., 2003; Albert, 2005) and, so, it was chosen to be used in this work. In this model, the connectivity is more likely in nodes with the higher number of connections, referencing the paradigm of 'the rich becomes more rich'. Specific details of the networks generated for this work are found in Section 5.3.

## 3 Differential evolution

The DE (Storn and Price, 1997) algorithm is a metaheuristic optimisation method based on vector operations. It has been used in many real-world hard optimisation problems (Shen and Zhang, 2015; Kalegari and Lopes, 2013; Krause and Lopes, 2013). In DE, a candidate solution for the optimisation problem is encoded as a real-valued vector, called individual $(\vec{x})$, of the dimension *NV*.

Similarly to other EC methods, DE holds a population of solutions (with *NP* individuals) that are evolved simultaneously throughout generations (or iterations). Consistently with the representation of individuals, specific operators (crossover and mutation) are used to generate new individuals in the population, by means of vector operations between them.

There are several different approaches for applying vector differences. One of them is randomly selecting three vectors, called parents $(\vec{x}_{r1}^G, \vec{x}_{r2}^G, \text{ and } \vec{x}_{r3}^G)$, where *r*1, *r*2 and *r*3 are different indices within the population of vectors, and *G* is the current generation of the population. Next, a child $\vec{x}_i^{G+1}$ is generated, where *i* is the *i*th individual.

Considering the single mutation operator, all *i*th elements in the target vector $\vec{x}_i^G$ are used for the generation of donor vectors $\vec{v}_i^G$, which are produced from the following operation:

$$\vec{v}_i^G = \vec{x}_{r1}^G + F\left(\vec{x}_{r2}^G - \vec{x}_{r3}^G\right) \tag{1}$$

where *F* is a scalar value in the range [0.4 … 1], which scales the vectorial difference of the vectors $\vec{x}_{r2}^G$ and $\vec{x}_{r3}^G$.

After the application of the mutation operator, the donor vector $\vec{v}_i^G$ exchange elements with the target vector $\vec{x}_i^G$, following a binomial model, defined by equation (2), which generates a test vector $\vec{u}_i^G = (u_1^G, ..., u_i^G)$.

$$u_i^G = \begin{cases} v_i^G, & \text{if } rand_i <= CR \text{ or } i = i_{rand}, \\ x_i^G, & \text{otherwise} \end{cases} \tag{2}$$

where $rand_i$ is a random value between [0, 1] and *CR* is the crossover probability defined between [0, 1]. The operation defines the value to be assigned to each element of the vector. This method ensures that individual $u_i^G$ will receive, at least, one position of the donor vector $\vec{v}_i^G$.

Next, the selection is done by an exhaustive criterion, in which individuals with lower values, in the case of minimisation problems, are included in the next generation (*G* + 1) population. The selection procedure is given by the equation (3).

$$x_i^{G+1} = \begin{cases} u_i^G, & \text{if } fitnessFunc\left(u_i^G\right) < fitnessFunc\left(x_i^G\right), \\ x_i^G, & \text{otherwise} \end{cases} \tag{3}$$

The discretised differential evolution (DDE) algorithm, proposed by Krause and Lopes (2013), is a variant of the original DE algorithm, and it was designed for working with binary vectors rather than real-coded vectors. The structure is exactly the same as the DE algorithm explained before. However, the value of each element in the vector, defined for continuous variables $x_i$, undergoes a discretisation procedure (explained in Section 5.1). The pseudo-code of DE is shown in Algorithm 1.

**Algorithm 1** Pseudo-code of the DE algorithm

```
Parameters: $NP, NV, CR, F$
Initialise the vector population $\vec{x}_i$
Compute fitnessFunc$(\vec{x}_i)$
while stopping criterion = FALSE do
    for $i \leftarrow 1$ to $NP$ do
        $\vec{v}_i^{G+1} = mutation\left(\vec{x}_i^G, F\right)$
        $\vec{u}_i^{G+1} = crossover\left(\vec{x}_i^G, \vec{v}_i^{G+1}, CR\right)$

    Compute fitnessFunc$(\vec{u})$
    for $i \leftarrow 1$ to $NP$ do
        if $fit_u > fit_x$ then
            $\vec{x}_i^{G+1} = \vec{u}_i^{G+1}$

        else
```

```
      $\vec{x}_i^{G+1} = \vec{x}_i^G$
  Find the current best $x_*$
```

*Source:* According to Krause and Lopes (2013)

## 4 SI methods

### 4.1 *Bat algorithm*

The BAT was originally proposed by Yang (2010) and it was inspired on the echolocation behaviour of bats with varying pulse rates of emission and loudness. Although it recently appeared, there are several applications of this algorithm (Cai et al., 2015; Parpinelli et al., 2014).

The search strategy of this algorithm is based on the elapsed time taken by the ultrasound waves (emitted by the bats) returning from the prey or an obstacle. The closer the object from the bat, the higher is the pulse rate. This allows the bat to evaluate the distance to the object. In the BAT, the quality (*fit*) of the position in the search space of each bat $(\vec{x})$ is measured by a fitness function (*fitnessFunc*), such that the best bat (best solution) is represented by $x_*$. The main parameters of the BAT are: pulse rate (*pr*) and loudness (*A*), which controls, respectively, the local and global search of the algorithm. Other two parameters, loudness decay factor ($\alpha$) and pulse increase factor ($\lambda$) are used to adjust the decay/increase the probability of local or global search, weighting the values of *pr* and *A*. The pseudo-code of the algorithm is shown in Algorithm 2.

**Algorithm 2** Pseudo-code of the BAT

```
Parameters: $NP$, $NV$, $\alpha$, $\lambda$
Initialise the bat population $\vec{x}_i$ and $\overrightarrow{vel}_i$ randomly
Define pulse frequency $\vec{f}_i$ at $\vec{x}_i$
for $i \leftarrow 1$ to $NP$ do
    Initialise pulse rates $pr_i$ and loudness $A_i$
end
Compute fitnessFunc$(\vec{x}_i)$
while stopping criterion = FALSE do
    for $i \leftarrow 1$ to $NP$ do
        Generate new solutions by adjusting:
        Frequency:
        $\vec{f}_i = \vec{f}_{mim} + (\vec{f}_{\max} - \vec{f}_{mim})\beta, \beta \in [0, 1]$
        Velocity:
        $\overrightarrow{vel}_i^G = \overrightarrow{vel}_i^{G-1} + (\vec{x}^G - \vec{x}_*)\vec{f}_i$
        Location:
        $\vec{x}^G = \vec{x}^{G-1} + \overrightarrow{vel}_i^G$
        if $rand > pr_i$ then
            Select a solution among the best solutions
            Generate a local solution around the selected best solution
        end
        Generate a new solution by flying randomly
        if $rand > A_i$ && $fitnessFunc(\vec{x}_i) < fitnessFunc(\vec{x}_*)$ then
            Accept the new solutions
            Increase $pr_i : pr_i^{G+1} = pr_i^0[1 - \exp(-\lambda G)]$
            Decrease $A_i : A_i^{G+1} = \alpha A_i$
        end
    end
    Find the current best $x_*$
end
```

*Source:* According to Parpinelli et al. (2014)

### 4.2 *Artificial bee colony*

The ABC algorithm is a SI optimisation algorithm created by Karaboga and Basturk (2008). ABC was inspired by the collective foraging behaviour of honey bees and it was shown to be efficient for interesting computational and engineering problems (Chidambaram et al., 2014; Wang et al., 2014).

**Algorithm 3** Pseudo-code of the ABC

```
Parameters: $NP$, $NV$, $limit$
Initialise the food sources $\vec{x}_i$ randomly
Compute fitnessFunc$(\vec{x}_i)$
$count_i = 0$
while stopping criterion = FALSE do
    Employed phase:
    for $i \leftarrow 1$ to $NP/2$ do
        Select k, j and r at random such that $k \in \{1, 2, \ldots, NP\}, j \in \{1, 2, \ldots, NV\}, r \in [0, 1]$
        $\vec{v} = \vec{x}_{ij} + r(\vec{x}_{ij} - \vec{x}_{kj})$
        Compute fitnessFunc$(\vec{v})$ and fitnessFunc$(\vec{x}_i)$
        if $fit_v$ is better than $fit_x$ then
            Greedy selection
        else
            $count_i = count_i + 1$
    Onlooker phase:
    for $i \leftarrow NP/2$ to $NP$ do
        Calculate selection probability
        Select a bee using the selection probability
        Produce a new solution $\vec{v}$ from the selected bee
        Compute fitnessFunc$(\vec{v})$ and fitnessFunc$(\vec{x}_i)$
        if $fit_v$ is better than $fit_x$ then
            Greedy selection
        else
            $count_i = count_i + 1$
    Scout phase:
    for $i \leftarrow 1$ to $NP$ do
        if $count_i > limit$ then
            $\vec{x}_i = random$
            $count_i = 0$
    Memorise the best solution achieved so far
```

*Source:* According to Parpinelli et al. (2014)

There are three types of bees in this algorithm: scout bees, employed bees and onlooker bees. Each type of bees browse the search space of the problem according to specific strategies. Food sources are positions of the search space where the amount of nectar represents the fitness or quality of the solution. The scout bees search the space randomly, so as to promote a global search. Each employed bee evaluates a food source and, later, transmits this information to other bees in the hive by means of a waggle dance. Onlooker bees are probabilistically influenced by this dance, which is proportional to the quality of solution, so that they will choose a promising food source to exploit. The pseudo-code of the ABC algorithm is shown in Algorithm 3.

## 5 Methods

In Section 5.1, we show how solutions are encoded and decoded for all the bioinspired methods previously presented in Sections 3 and 4. Next, Section 5.2 details the fitness function used in this work. The control parameters of the optimisation methods are shown in Section 5.3. Finally, the parameters of the AGN, as well as the analysis of the temporal data of the gene expression are presented in Section 5.4.

### 5.1 Encoding and decoding

In a previous work (Hattori et al., 2015), possible solutions to the GRN problem were encoded in a vector of real numbers and later discretised (see below the decoding procedure). The $i^{\text{th}}$ element of the discretised vector represents the presence (1) or absence (0) of the $i^{\text{th}}$ gene as a predictor for a given target gene. In the current work, we used a more efficient approach in which sets of bits of the vector encodes an index. This index, in turn, represents a given predictor gene. Also, based on biological evidences, the maximum number of predictors for a given target was limited to 4. An extra bit was included with each index to indicate whether or not it is activated. Therefore, it is possible to find from 0 to 4 predictors for each target.

All EC and SI methods used in this work were originally devised to tackle with continuous search spaces. That is, the vector that encodes a possible solution for the problem is a string of real numbers. However, as mentioned before, the encoding used here requires binary numbers to construct indices to predictor genes. Therefore, a specific decoding procedure was devised. In Figure 3, a real-valued vector $\vec{x}_i$ in the range [–1 … 1] represents a candidate solution. Each element of the vector is applied to a sigmoid function [equation (4)], so that it is converted to a bit $\{0, 1\}$. A set of bits thus encodes an index that can be activated or not, depending on the extra bit.

$$\vec{x}_i = \begin{cases} 1, & \text{if } \dfrac{2}{1+e^{-2x_i}} - 1 > 0, \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\vec{x}_i$ is an element of the discretised vector.

This procedure was shown to be more efficient than a pure binary encoding (Krause and Lopes, 2013; Krause et al., 2013), since algorithms work in a smooth and continuous search space, so that values evolve gradually, without abrupt changes (as with binary numbers). However, ultimately, the algorithm deals with discrete problems.

### 5.2 Fitness function

The conditional entropy $H(Y \mid x)$ represents a measure of uncertainty associated with a random variable $Y$, given that the value of a second random variable $x$ is known. In other words, the lower the conditional entropy of $Y$ given $x$, the better will be the prediction of $Y$'s behaviour by observing the variable $x$. Conditional entropy is defined as:

$$H(Y \mid x) = -\sum_{y \in Y} P(y \mid x) \log P(y \mid x). \tag{5}$$

Furthermore, the MCE (Lopes et al., 2008) is defined by the weighted average of the conditional entropy of all possible instances $x \in X$, given by:

$$H(Y \mid X) = \frac{1}{n} \sum_{x \in X} P(x) H(Y \mid x), \tag{6}$$

where $n$ is the number of binary states of $X$.

The lower the $H(Y \mid X)$ value, the greater the gain of information about $Y$ by the observation of $X$.

In this context, the inference of GRNs is performed by considering the conditional dependence of a target gene $Y$ given a potential subset of predictors $X$ from their expression profiles, and applying the MCE as criterion function. Several methods for the inference of GRNs based on this measure were proposed in the literature (Barrera et al., 2007; Vicente et al., 2012; Jimenez et al., 2015; Lopes et al., 2014a). Therefore, it has been recognised as an appropriate statistical tool to model direct interactions between genes.

**Figure 3** Decoding procedure from the encoded vector $\vec{x}_i$ to the indices of genes (features)

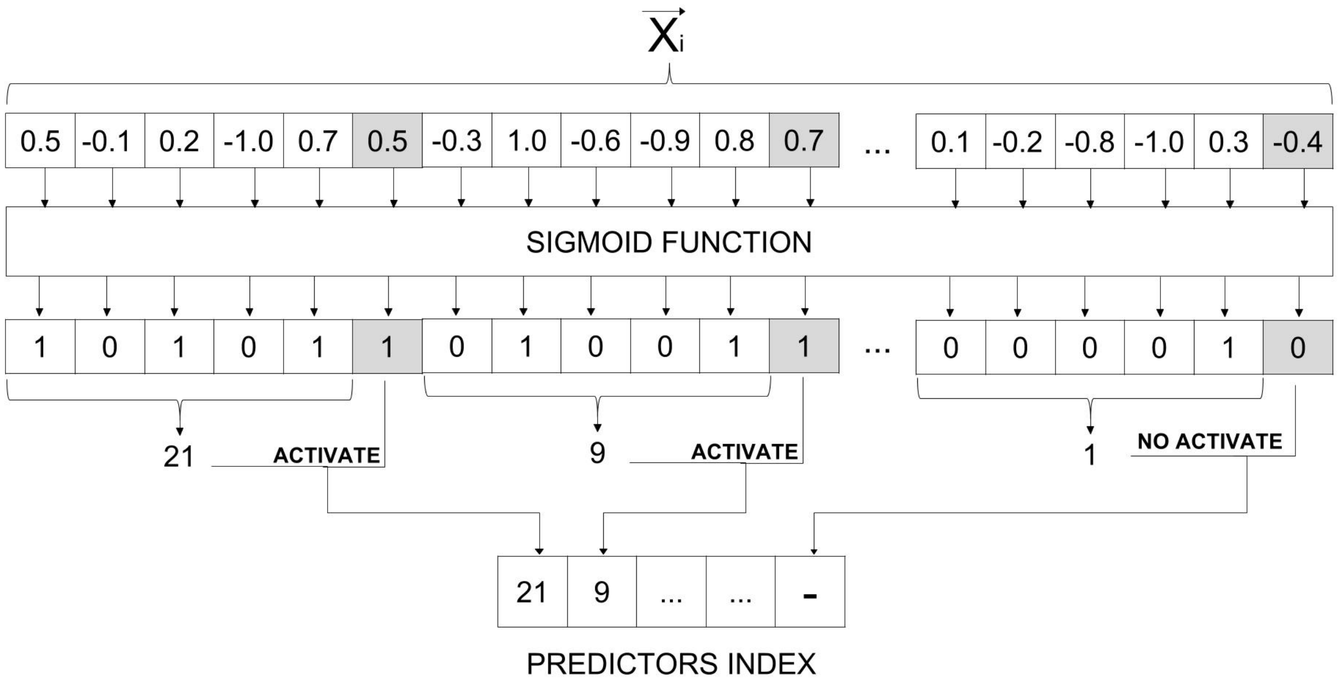
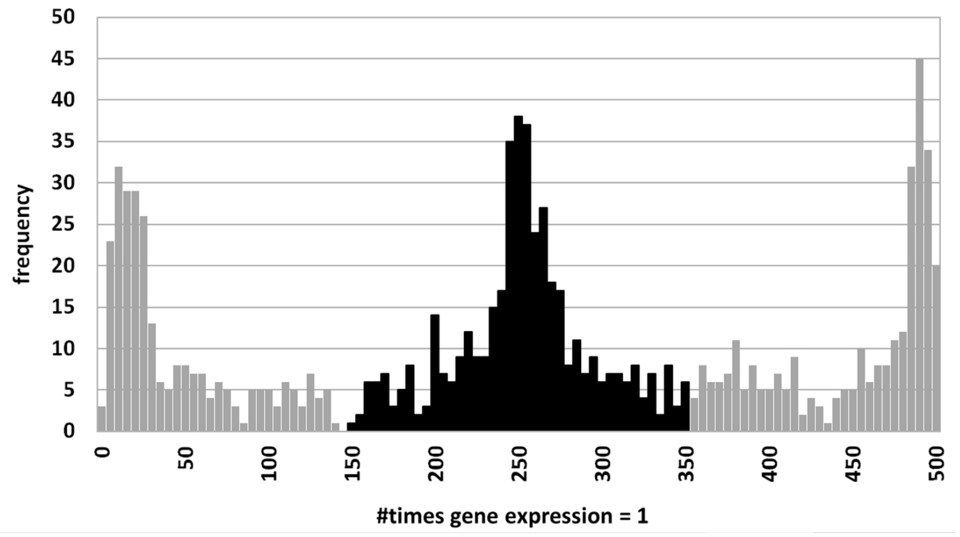


**Figure 4** Histogram showing the distribution of number of times genes are expressed



Note: Genes in the extremities tend to be most times in 0 or 1, suggesting they are not appropriate for the inference of GRNs.

## 5.3 Control parameters

For all algorithms, DE, BAT and ABC, we used the standard control parameters as long as possible. The number of iterations (generations) for each run was set to a very large number (50,000) so as to allow enough iterations and guarantee the full convergence of the algorithm. This procedure allows a further analysis of number of iterations needed for the algorithm to converge.

Since all algorithms are population-based, we established the same value for all of them ($NP$ = 100) so that, considering the number of iterations, the computational effort of all algorithms is exactly the same (100 × 50,000 = 5 × $10^6$ fitness evaluations. The remaining parameters of the algorithms are:

- DE: crossover rate ($CR$ = 0.8), mutation rate ($F$ = 0.05)
- BAT: loudness decay factor ($\alpha$ = 0.9), pulse increase factor ($\lambda$ = 0.9)

- ABC: number of the employed bee and onlooker bees (*employedbee* = 50, *onlookerbees* = 50), parameter for activating search as scout bees (*limits* = 100).

### 5.4 Generation and evaluation of GRNs

As mentioned before, we used the scale-free (BA) model for generating the AGNs (see Section 2.2), since it is usual for biological networks. The average number of connections between predictors and a given target gene was set to ($\langle k \rangle$ = 3), based on Kauffman (1993) who stated that such value remains in the border between order and chaos.

Based on the work of Marbach et al. (2012), considered a gold standard in the inference of GRNs, we established in this study a total number of 1,000 genes for which 500 gene expressions were generated from an initial stimulus. The network dynamics was generated from probabilistic Boolean functions (Shmulevich et al., 2002), generating a simulated expression signal, which can be used as input for the network inference method.

More specifically, the temporal expression data is given by a matrix, in which its rows are the genes and the columns are their expression data. Each column stores the expression of all genes in a given time $t$. Therefore, the expression data of a gene at time $t + 1$ is given by a logic circuit using the values of its predictors genes in time $t$. Performing this step for all genes, the expression data in time $t + 1$ is complete and can be repeated according to the desired signal size.

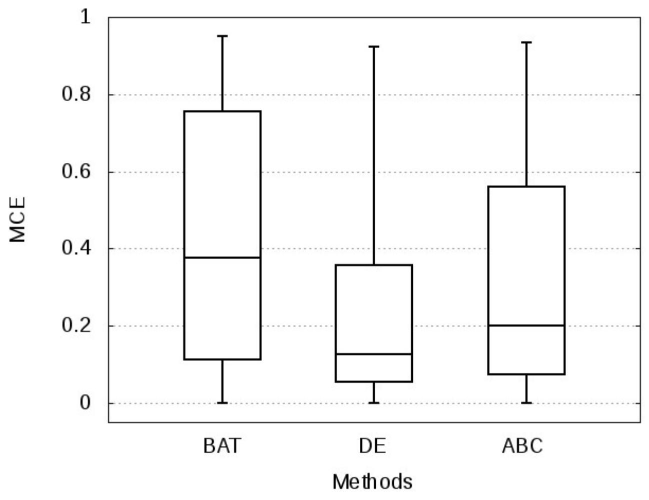
A preprocessing in the expression profiles was found necessary. Since data was randomly generated, it is possible that a given gene has almost any expression profile along time. At the extremes, a gene can be always activated or always inhibited, with all possible variability in between. Therefore, genes with small variations in the expression profile are useless for the simulation and inference of GRNs. Figure 4 shows a histogram of the distribution of the number of times genes are expressed. For the purposes of this work, we established upper (350) and lower bounds (150) for the number of times a gene could be expressed. These limits leaded to 315 (out of 1,000) target genes to be studied in this work (black lines in the figure).

## 6 Experiments and results

In this section, the results of our experiments are presented, focusing the comparison of performance among EC (DE) and SI (BAT and ABC) methods. All the algorithms tested were implemented in ANSI-C programming language and run in a cluster of PCs running Ubuntu Linux. Due to the stochastic nature of the EC and SI algorithms, they were run 20 times (with different initial random seeds) for each of the 315 optimisation problems. Considering the computational effort requested by the algorithms mentioned in Section 5.3, and the number of problems and repetitions, each algorithm was run 6,300 times, and the total number of fitness evaluations was $94.5 \times 10^9$.

Results of the runs for each algorithm are shown in the box plot of Figure 5. Each box plot accounts for the best MCE values found in all executions for all target genes, showing the maximum, minimum, median and upper and lower quartiles. It is possible to observe that no algorithm is definitely better than the other and all them have a reasonable large variability, ranging from one extreme to the another. The largest variability was observed for the BAT and the smallest one for the DE. Also, the median indicates that DE has achieved better MCE values most frequently than the other algorithms.

**Figure 5** Box plot of MCE for each algorithm (BAT, DE, ABC), considering all executions and all target genes



Regarding the convergence of the algorithms, Figure 6 shows a sample of the evolution of the best fitness found by each algorithm along the 50,000 iterations. Each point of the curves is the average of 20 executions. In the figure, data were collected for the target gene #6, and a similar behaviour was also observed for the other genes. Since the convergence occurred soon before the 100th iteration for BAT and DE, a zoom of the first 300 iterations is shown in the top left of the figure. This was due to the running parameters used in BAT and DE.

It is clear from Figure 6 that ABC converged much slower than the other algorithms, indicating its ability to maintain global search for more time than the other. On the other hand, DE consistently converged faster and, in general, to better results than the other algorithms. Possibly, this fact accounted for the smaller dispersion of results and median best values, as seen in Figure 5.

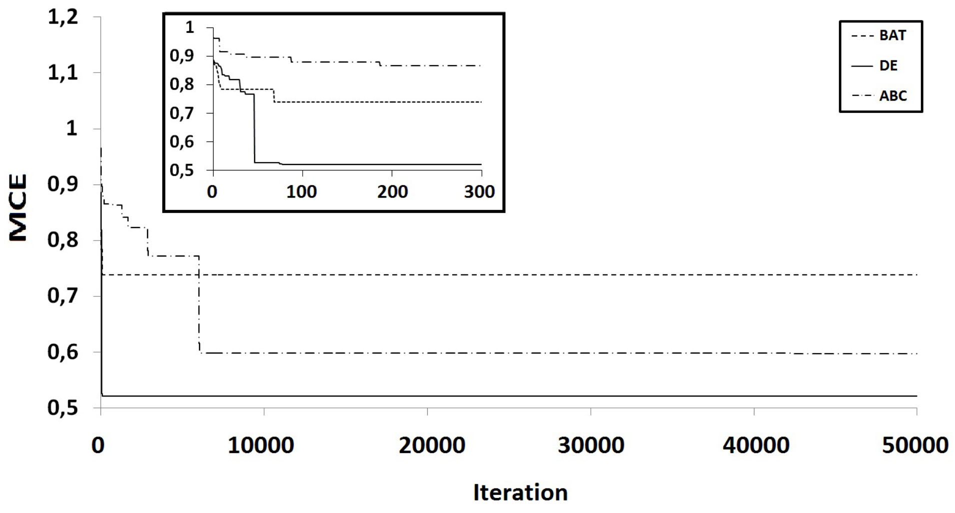
Another way to compare algorithms is verifying their success rate, calculated as follows. For a given problem (in this case the inference of the GRN for a specific target gene), an algorithm is considered successful if the best MCE found among the 20 executions is under a threshold $\xi$ = 0.3. This is the default parameter suggested for the method proposed by Lopes et al. (2008). Considering that more than one algorithm can achieve MCE values under the established threshold in some run, the overall success rate is high and quite similar for all algorithms, as shown in Figure 7. The light bars show the proportion of the 315 problems in which each algorithm achieved success. In this case, DE was a little better than the other. Now, considering only those problems in which some algorithm obtained success in any run (213 out of 315), and then comparing which one obtained the best value, the dark bars show that DE, again, was the best-performing algorithm.

Finally, we examined in deep the gene sets found in the best run of the algorithms and compared to each other. Similarly, we compared the best gene set found by the algorithms with the original gene set created with the scale-free BA model. The measure used is the average Jaccard (1912) similarity coefficient, shown in equation (7), and taking the average along all cases (target genes), see Figure 8.

$$\bar{J}(A, B) = \frac{1}{n}\sum_{i=1}^{n} \frac{|A_i \cap B_i|}{|A_i \cup B_i|} \tag{7}$$

where $A_i$ and $B_i$ are the gene sets found by methods $A$ and $B$ for the $i^{\text{th}}$ target gene. Since this measure is the intersection of two sets divided by the union of them, it gives the proportion of common elements between the two sets.

**Figure 6** Evolution of the average fitness for all algorithms (BAT, DE and ABC) considering the search for predictor genes for the target gene #6



Note: At the top left it is the zoom of the first 300 iterations.

**Figure 7** Performance comparison of BA, DE and ABC, regarding the success rate

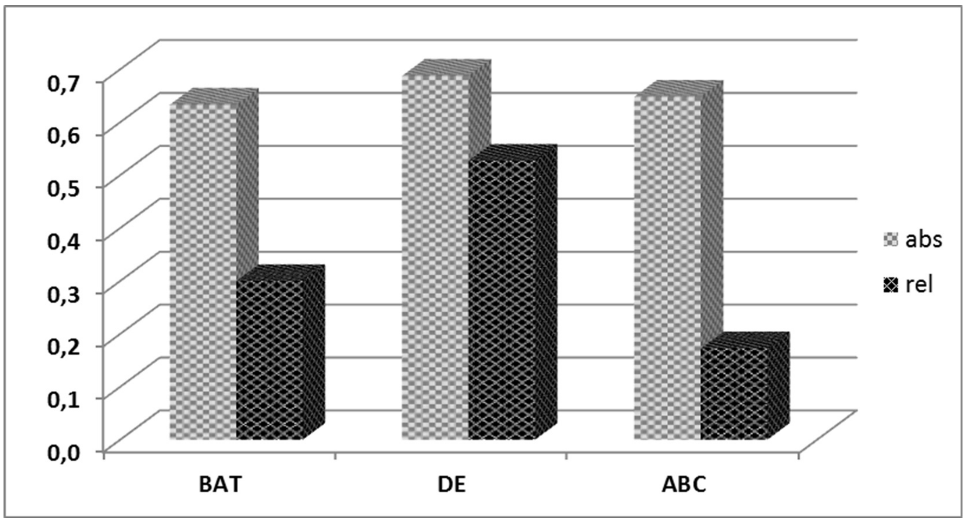
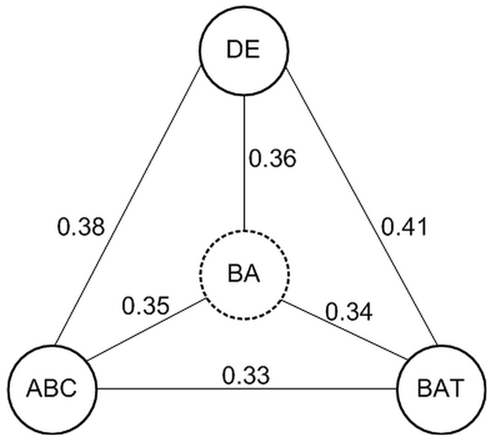


**Figure 8** Average Jaccard similarity coefficient between the algorithms and between them and the original scale-free BA model



The interpretation of the figure suggests that all the three methods found gene sets with about the same similarity with the original predictors created with the scale-free BA model. However, such similarity is rather low (~0.35) suggesting that these methods are capable of finding partial solutions for the inference of genes. Notwithstanding, in general, the box plots for DE and ABC showed that best solutions found are rather good. Figure 8 shows that the pair of methods that obtained the most similar gene sets were DE and BAT (in comparison with DE and ABC, as well as with BAT and ABC).

## 7 Conclusions and future work

In this work, we analysed the performance of EC (DE) and SI (BAT and ABC) methods for the inference of GRNs from gene expression profiles. This was done using an AGN based on the scale-free (BA) topology. An unbiased comparison of the standard versions of the algorithms was done, taking into account not only the quality of solutions but, also, the success rate and the similarity between the original gene set and the gene sets inferred by the methods.

The experimental results indicate that all methods presented high variability. However, it is possible to notice that DE algorithm showed a smaller variation among the algorithms tested. Additionally, the DE algorithm achieved the best results, presenting more frequently small MCE values than the other methods which, in general, leads to better predictors genes.

Moreover, when the performance of the algorithms is taking into account, DE again achieved the best success rate. Also, this algorithm was the one that found gene sets more similar to the original gene network. Overall, in all analyses performed the DE algorithm showed the best results when compared to BAT and ABC algorithms.

To foster the most fair comparison between algorithms, no effort was done to optimise their running parameters, and we used their default ones. Of course, it is possible that the results obtained could be improved if a fine-tuning of parameters would be done, for each particular optimisation problem. However, this is beyond the scope of this paper. Anyhow, the overall analysis of results suggests that DE is the most promising algorithm for the problem, since it consistently achieved good results. Therefore, improvements in the DE algorithm can be a starting point for future work, with special attention to hybridisation, since it has been shown very efficient for hard optimisation problems (Parpinelli and Lopes, 2015).

Also, further work will include more experiments regarding to the sensitivity of the algorithms to changes in the parameters of the GRNs, such as the number of genes in the network, the average number of connections k and the number of temporal expressions. It is also intended to do experiments with biological data, such as in Marbach et al. (2012), so as to evaluate the main differences between simulated and real-world data.

## Acknowledgements

This work was partially supported by grants from CNPq (to H.S. Lopes and F.M. Lopes), CAPES (to L.R. Hattori) and Fundação Araucária (to F.M. Lopes).

## References

Alba, E., García-Nieto, J., Jourdan, L. and Talbi, E-G. (2007) 'Gene selection in cancer classification using PSO/SVM and GA/SVM hybrid algorithms', in *IEEE Congress on Evolutionary Computation*, IEEE Press, Piscataway, NJ, pp.284–290.

Albert, R. (2005) 'Scale-free networks in cell biology', *Journal of Cell Science*, Vol. 118, No. 21, pp.4947–4957.

Barabási, A-L. and Albert, R. (1999) 'Emergence of scaling in random networks', *Science*, Vol. 286, No. 5439, pp.509–512.

Barrera, J., Cesar Jr., R.M., Martins Jr., D.C., Vencio, R.Z.N., Merino, E.F., Yamamoto, M.M., Leonardi, F.G., Pereira, C.A.B. and Portillo, H.A. (2007) 'Constructing probabilistic genetic networks of plasmodium falciparum from dynamical expression signals of the intraerythrocytic development cycle', in *Methods of Microarray Data Analysis*, Vol. 5, pp.11–26, Springer, Boston, MA, USA.

Byrne, J., Nicolau, M., Brabazon, A. and O'Neill, M. (2014) 'An examination of synchronisation in artificial gene regulatory networks', in *IEEE Congress on Evolutionary Computation (CEC)*, IEEE Press, Piscataway, NJ, pp.2764–2769.

Cai, X., Li, W., Kang, Q., Wang, L. and Wu, Q. (2015) 'Bat algorithm with oscillation element', *International Journal of Innovative Computing and Applications*, Vol. 6, No. 3, pp.171–180.

Chandra, N. and Padiadpu, J. (2013) 'Network approaches to drug discovery', *Expert Opinion on Drug Discovery*, Vol. 8, No. 1, pp.7–20.

Chidambaram, C., Vieira Neto, H., Dorini, L.B. and Lopes, H.S. (2014) 'Multiple face recognition using local features and swarm intelligence', *Information and Communication Engineers Transactions in Information Systems*, Vol. E97-D, No. 6, pp.1614–1623.

Das, S., Abraham, A. and Konar, A. (2008) 'Swarm intelligence algorithms in bioinformatics', in *Computational Intelligence in Bioinformatics*, Springer, Heidelberg, Germany, pp.113–147.

Erdös, P. and Rényi, A. (1959) 'On random graphs', *Publicationes Mathematicae Debrecen*, Vol. 6, No. 1, pp.290–297.

Hattori, L.T., Lopes, H.S. and Lopes, F.M. (2015) 'A discretized differential evolution algorithm for the inference of gene regulatory networks', in *Latin America Congress on Computational Intelligence (LA-CCI)*, IEEE Press, pp.1–6.

Jaccard, P. (1912) 'The distribution of the flora in the alpine zone', *New Phytologist*, Vol. 11, No. 2, pp.37–50.

Jimenez, R.D., Martins Jr., D.C. and Santos, C.S. (2015) 'One genetic algorithm per gene to infer gene networks from expression data', *Network Modeling Analysis in Health Informatics and Bioinformatics*, Vol. 4, No. 1, pp.1–22.

Kalegari, D.H. and Lopes, H.S. (2013) 'An improved parallel differential evolution approach for protein structure prediction using both 2D and 3D off-lattice models', in *IEEE Symposium on Differential Evolution*, IEEE Press, Piscataway, NJ, pp.143–150.

Karaboga, D. and Basturk, B. (2008) 'On the performance of artificial bee colony (ABC) algorithm', *Applied Soft Computing*, Vol. 8, No. 1, pp.687–697.

Kauffman, S.A. (1993) *The Origins of Order: Selforganization and Selection in Evolution*, Oxford University Press, Oxford.

Kelemen, A., Kelemen, A., Abraham, A. and Chen, Y. (2008) *Computational Intelligence in Bioinformatics*, 1st ed., Springer Publishing Company, NY, USA.

Krause, J. and Lopes, H.S. (2013) 'A comparison of differential evolution algorithm with binary and continuous encoding for the MKP', in *BRICS Congress on Computational Intelligence*, IEEE Computer Society, Washington, DC, pp.381–387.

Krause, J., Cordeiro, J.A., Parpinelli, R.S. and Lopes, H.S. (2013) 'A survey of swarm algorithms applied to discrete optimization problems', in Yang, X-S., Cui, Z., Xiao, R., Gandomi, A.H. and Karamanoglu, M. (Eds.): *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, pp.169–191, Elsevier Science, Amsterdam, The Netherlands.

Lopes, F.M., Cesar Jr., R.M. and Costa, L.F. (2011a) 'Gene expression complex networks: synthesis, identification, and analysis', *Journal of Computational Biology*, Vol. 18, No. 10, pp.1353–1367.

Lopes, F.M., de Oliveira, E.A. and Cesar Jr., R.M. (2011b) 'Inference of gene regulatory networks from time series by Tsallis entropy', *BMC Systems Biology*, Vol. 5, No. 1, pp.1–13.

Lopes, F.M., Martins Jr., D.C., Barrera, J. and Cesar Jr., R.M. (2014a) 'A feature selection technique for inference of graphs from their known topological properties: revealing scale-free gene regulatory networks', *Information Sciences*, Vol. 272, No. 1, pp.1–15.

Lopes, F.M., Ray, S.S., Hashimoto, R.F. and Cesar Jr., R.M. (2014b) 'Entropic biological score: a cell cycle investigation for GRNs inference', *Gene*, Vol. 541, No. 2, pp.129–137.

Lopes, F.M., Martins, D.C. and Cesar Jr., R.M. (2009) 'Comparative study of GRNs inference methods based on feature selection by mutual information', in *IEEE International Workshop on Genomic Signal Processing and Statistics*, IEEE Press, Piscataway, NJ, pp.1–4.

Lopes, F.M., Martins, D.C. and Cesar, R.M. (2008) 'Feature selection environment for genomic applications', *BMC Bioinformatics*, Vol. 9, No. 1, p.451.

Marbach, D., Costello, J.C., Küffner, R., Vega, N.M., Prill, R.J., Camacho, D.M., Allison, K.R., Kellis, M., Collins, J.J., Stolovitzky, G. et al. (2012) 'Wisdom of crowds for robust gene network inference', *Nature Methods*, Vol. 9, No. 8, pp.796–804.

Parpinelli, R.S. and Lopes, H.S. (2015) 'A computational ecosystem for optimization: review and perspectives for future research', *Memetic Computing*, Vol. 7, No. 1, pp.28–41.

Parpinelli, R.S., Benítez, C.M.V., Cordeiro, J. and Lopes, H.S. (2014) 'Performance analysis of swarm intelligence algorithms for the 3D-AB off-lattice protein folding problem, *Journal of Multi-Valued Logic and Soft Computing*, Vol. 22, No. 3, pp.267–286.

Shen, X. and Zhang, M. (2015) 'A differential evolution-based memetic algorithm for project scheduling problems', *International Journal of Innovative Computing and Applications*, Vol. 6, No. 3, pp.229–239.

Shmulevich, I. and Dougherty, E.R. (2007) *Genomic Signal Processing*, Princeton University Press, Princeton, NJ.

Shmulevich, I., Dougherty, E.R. and Zhang, W. (2002) 'From Boolean to probabilistic Boolean networks as models of genetic regulatory networks', *Proceedings of the IEEE*, Vol. 90, No. 11, pp.1778–1792.

Storn, R. and Price, K. (1997) 'Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces', *Journal of Global Optimization*, Vol. 11, No. 4, pp.341–359.

Stuart, J.M., Segal, E., Koller, D. and Kim, S.K. (2003) 'A gene-coexpression network for global discovery of conserved genetic modules', *Science*, Vol. 302, No. 5643, pp.249–255.

Terfve, C., Cokelaer, T., Henriques, D., MacNamara, A., Gonçalves, E., Morris, M.K., van Iersel, M., Lauffenburger, D.A. and Saez-Rodriguez, J. (2012) 'Cellnoptr: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms', *BMC Systems Biology*, Vol. 6, No. 133, pp.1–14.

Vicente, F.F.R., Lopes, F.M., Hashimoto, R.F. and Cesar Jr., R.M. (2012) 'Assessing the gain of biological data integration in gene networks inference', *BMC Genomics*, Vol. 13, No. 6, pp.1–12.

Wang, H., Wang, W., Zhao, J., Zhu, H. and Sun, H. (2014) 'A hybrid artificial bee colony based on differential evolution for production scheduling problems', *International Journal of Innovative Computing and Applications*, Vol. 6, No. 2, pp.55–62.

Wang, Z., Gerstein, M. and Snyder, M. (2009) 'RNA-Seq: a revolutionary tool for transcriptomics', *Nature Reviews Genetics*, Vol. 10, No. 1, pp.57–63.

Watts, D.J. and Strogatz, S.H. (1998) 'Collective dynamics of small-world networks', *Nature*, Vol. 393, No. 6684, pp.440–442.

Yang, X-S. (2010) 'A new metaheuristic bat-inspired algorithm', in Cruz, C., González, J.R., Krasnogor, N. Pelta, D.A. and Terrazas, G. (Eds.): *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, pp.65–74, Springer, Heidelberg, Germany.