GAIS: A Gaussian Artificial Immune System for Continuous Optimization

Pablo A.D. Castro and Fernando J. Von Zuben

Laboratory of Bioinformatics and Bioinpired Computing (LBiC),
Department of Computer Engineering and Industrial Automation (DCA),
School of Electrical and Computer Engineering (FEEC),
University of Campinas (Unicamp),
P.O. Box 6101, 13083-852 Campinas-SP, Brazil
{pablo,vonzuben}@dca.fee.unicamp.br

Abstract. This paper proposes a Gaussian Artificial Immune System (GAIS) to deal effectively with building blocks (high-quality partial solutions coded in the solution vector) in continuous optimization problems. By replacing the mutation and cloning operators with a probabilistic model, more specifically a Gaussian network representing the joint distribution of promising solutions, GAIS takes into account the relationships among the variables of the problem, avoiding the disruption of already obtained high-quality partial solutions. Two versions of the algorithm were developed. In the first one, the estimation of the joint probability distribution is achieved by means of a single multivariate Gaussian distribution. In the second version, the estimation is carried out using a Gaussian mixture model. The algorithms were applied to eight benchmarks and the results compared with those produced by an immune-inspired algorithm and an estimation of distribution algorithm.

Keywords: Artificial immune system, Gaussian network, Gaussian mixture model, continuous optimization.

1 Introduction

Over the last decades, a variety of bio-inspired algorithms have been proposed for solving optimization problems. Among the appealing approaches, artificial immune systems (AISs) have received special attention due to their interesting features: (*i*) dynamic control of population size in response to the particularities of the problem; (*ii*) efficient mechanism of exploration/exploitation of the search space, which allows to find and preserve the local optima as well as to insert and maintain diversity in the population [1] [2] [3].

Despite their high performance as general problem solving tool, there are some shortcomings associated with these immune-inspired algorithms. Firstly, as the complexity and scale of the problem increase, the performance of the algorithms becomes more and more associated with a proper choice of the design parameters, such as mutation rate. Otherwise, very poor solutions can be generated [4]. In addition, it is noticeable that, when the solution is represented by a vector of attributes, the population of candidate solutions may contain partial high-quality solutions to the problem, called building blocks [5]. The existing AISs suffer from the lack of ability to identify and effectively manipulate building blocks of the problem [6]. As affinity maturation requires cloning followed by the mutation of the newly-generated cells, and assuming that the mutation operator cannot discover by itself crucial relationships among the variables of the problem, building blocks are not supposed to survive, being disrupted by mutation.

Recently, we have proposed an immune-inspired algorithm for solving combinatorial optimization problems, denoted Bayesian Artificial Immune System (BAIS) [6] which replaces the traditional mutation operator with a probabilistic model representing the probability distribution of the promising solutions found so far. Then the obtained probabilistic model is used to generate new individuals. A Bayesian network was adopted as the probabilistic model, due to its capability to properly capture the most expressive interactions among the variables of the problem. Besides the capability to deal with building blocks, BAIS still preserves the aforementioned advantages of AISs. The proposed algorithm was successfully applied to many optimization problems and the results are reported in the literature [7] [8] [9] [10].

Now, we extend the proposal in [6] aiming at investigating its usefulness in continuous optimization problems. Since the variables of the problem are continuous, the probabilistic model utilized is a Gaussian network, guiding to Gaussian Artificial Immune Systems (GAISs). Two versions of the algorithm were developed. The first one utilizes a single multivariate Gaussian probability distribution and the other one utilizes a Gaussian mixture model. In order to alleviate the computational cost required to learn the structure of the Gaussian network, this task is performed at a certain number of iteration while the probabilities associated with the structure are updated at each iteration.

The main objective of this study is to design a competent algorithm with qualitative advantages over the contenders, as will be outlined in Section 4. Generally, the corresponding quantitative advantages arise as a natural consequence. Experiments on eight well-known functions have been carried out to evaluate the effectiveness of the proposed methodology when compared to other algorithms.

This paper is organized as follows. In Section 2, we provide a background to artificial immune system and its limitation regarding the handling of building blocks. Section 3 describes the GAIS in details. The experimental results are outlined and analyzed in Section 4. Finally, in Section 5 we draw some concluding remarks and present the further steps of the research.

2 Artificial Immune Systems and Building Blocks

Artificial Immune System (AIS) is a relative new computational paradigm inspired by the immunological system of vertebrates and designed for solving a wide range of problems, such as optimization, clustering, pattern classification and computational security [2] [11].

Several immune-inspired algorithms have been proposed in the literature, and they differ mainly on the immunological metaphor used as inspiration. Two important principles from immunology are the Clonal Selection Theory [12] and the Immune Network Theory [13]. Every standard AIS based on these two principles evolves the population of solutions according to their fitness by increasing the concentration of particular antibodies. The newly-generated antibodies are mutated, and antibodies with the lowest fitness among similar antibodies are suppressed. As the complexity and scale of the problem handled by the algorithm increase, the performance becomes more and more associated with a proper choice of the design parameters [4].

Moreover, since many problems can be decomposed into sub-problems, it is noticeable that antibodies in the population may contain partial solutions to the global problem. These partial solutions are called building blocks [5]. The existing artificial immune systems suffer from the lack of ability to identify and manipulate these interactions among the variables of the problem. Traditional mutation operators generally adopted in the literature can easily guide to the disruption of these partial solutions.

In the bio-inspired computation area, various attempts to prevent the disruption of important partial solutions have been made by changing the representation of solutions in the algorithm, or by designing specific operators [5] [14] [15] [16]. However, these approaches are highly problem-dependent, leading to the design of very specific strategies. Furthermore, they require prior domain knowledge of the problem so that the variables in the antibodies can be properly arranged with respect to the mutation operator. Actually, in most of the real-world problems, this knowledge is not available a priori.

One way to allow AISs to handle building blocks effectively is to make the algorithm learn the relationships among the variables by using statistical information extracted from the set of promising solutions. From a conceptual point of view, the selected set of promising solutions can be viewed as a sample drawn from an unknown probability distribution. An estimation of this probability distribution would allow the immune algorithm to generate new solutions that are somehow similar to the ones contained in the original selected solutions. Additionally, a probability distribution can effectively capture a building block structure of a problem, even without any prior information. Under these assumptions, challenging optimization problems with complex interactions among the variables may then become more manageable.

This interesting approach to deal with building blocks is inspired by a class of evolutionary algorithms denoted Estimation of Distribution Algorithms (EDAs) [17] [18]. These evolutionary algorithms replace the traditional crossover and mutation operators with a probabilistic model. This probabilistic model represents the distribution of probabilities of the promising solutions and can be used to generate new individuals.

There are several EDAs in the literature with alternative probabilistic models, depending on the intended degree of relationship among the variables [19] [20]. Despite the appeal of EDAs and the reported success, there are some difficulties in applying them effectively. One difficulty is related to the lack of diversity in the population [21]. Since EDAs generate new solutions based on the distribution of probability of previous selected solutions, they tend to explore the search space in a biased way. It is more likely to visit a region of the search space already visited in the past, leading the algorithm to get trapped into local minima easily. As a result, any basic EDA is not capable of dealing with multimodal optimization problems in an effective way. Although additional mechanisms for diversity maintenance can be incorporated into EDAs, they still tend to present a poorer performance when compared with artificial immune systems, which are algorithms inherently designed to promote diversity [22]. This aspect constitutes one of the motivations to develop our algorithms, and one of the arguments to support the advantage of our proposals over EDAs.

3 Gaussian Artificial Immune System

We propose a novel immune-inspired algorithm for solving continuous optimization problems which has the mutation and cloning operators replaced by a probabilistic model in order to generate new antibodies. The probabilistic model used in our proposal is a Gaussian network due to its capability to properly represent the most expressive interactions among the variables.

The pseudo-code of the proposed algorithm, called Gaussian Artificial Immune System (GAIS), is presented in Algorithm 1. Notice that the cloning and mutation steps were replaced with the building of the Gaussian network and the subsequent sampling of new individuals according to the generated model.

Algorithm 1. Gaussian Artificial Immune System

```
Begin
 Initialize the population of antibodies;
while stopping condition is not met do
   Evaluate the population;
   Select the best antibodies;
   Build the Gaussian network at every $p$ iterations;
   Sample new antibodies;
   Suppress antibodies with fitness lower than a threshold;
   Eliminate similar antibodies;
   Insert new antibodies randomly;
end while
End
```

In GAIS, the initial population is generated at random. From the current population, the best solutions are selected. A Gaussian network that properly fits the selected antibodies is constructed. A number of new antibodies sampled from the network are then inserted into the population and those similar ones and with lower fitness are eliminated. We computed similarity based on the vector of solutions. Next, few individuals are generated randomly and inserted into the population in order to favour diversity, yielding a better exploration of the search space.

Aiming at decreasing the computational cost of the algorithm and looking for a speed up in the execution time, the structure of the network is rebuilt at every p iterations ($p > 1$), whereas the redefinition of the probabilities remains at every iteration.

Besides the capability to maintain diversity in the population, GAIS also performs multimodal optimization, adjusts dynamically the size of the population according to the problem, and, most importantly, identifies and manipulates building blocks. As a direct consequence of these characteristics, the search process becomes more robust in the sense that there is no much variation in performance when the algorithm is run several time for the same problem. In addition, a considerable reduction in the number of evaluations of candidate solutions occurs until the algorithm reaches a certain level of performance.

Two aspects should receive special attention. The first is related to the way of building the Gaussian network from the selected individuals; and the other is how to use the network to generate new solutions. In the next section we explain in detail how to perform these two tasks.

3.1 Gaussian Network - Learning and Sampling

These two tasks, learning and sampling of Gaussian network, are fundamental to the GAIS algorithm. Learning the Gaussian network for a given set of promising solutions corresponds to estimating their joint distribution. Sampling new instances according to the network guides to new candidate solutions to the problem.

Gaussian networks are probabilistic graphical models often used to estimate probabilistic dependencies among variables belonging to continuous domains [23]. Formally, a Gaussian network for a set of variables $X = \{X_1, X_2, ..., X_n\}$ is a directed acyclic graph whose nodes are variables of the problem and the edges indicate relationships of dependence among the connected variables. The Gaussian network receives this name because the joint probability for X is represented by an n-dimensional Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$:

$$f(x) = f(x_1 x_2 \ldots x_n) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-n/2} |\boldsymbol{\Sigma}|^{-1/2} e^{-1/2(x-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})}, \quad (1)$$

where $\boldsymbol{\Sigma}$ is the determinant of the covariance matrix and $\boldsymbol{\Sigma^{-1}}$ is the inverse of the covariance matrix, also called precision matrix and denoted by W.

For instance, if there is an edge from node X_1 to node X_2, we say that variable X_1 is parent of variable X_2 and, therefore, the value of X_2 is in a conditional dependence on the value of X_1. If a variable X_i has a set of parents, denoted here by $\boldsymbol{pa_i}$, its probability distribution is characterized by a conditional probability density function (pdf) and it is expressed by $f(X_i|\boldsymbol{pa_i})$. On the other hand, the probability distribution of a variable X_j which has no parents is expressed by its unconditional pdf, $f(X_j)$. In this sense, the joint probability distribution for X can be re-written as a product of conditional pdf, each of which belongs to an independent Gaussian distribution. That is:

$$f(x) = \prod_{i=1}^{n} f(x_i|\boldsymbol{pa_i}), \tag{2}$$

where

$$f(x_i|\boldsymbol{pa_i}) = \mathcal{N}(\mu_i + \sum_{x_k \in \boldsymbol{pa_i}} b_{ki}(x_k - \mu_k), \sigma_i^2), \tag{3}$$

where μ_i is the mean of X_i, σ_i^2 is the variance of X_i conditioned to the parents of X_i and b_{ki} is a linear coefficient reflecting the strength of the relationship between X_k and X_i. If $b_{ki} = 0$, then there is no relationship between the variables X_k and X_i.

The transformation of b_{ki} and σ_i^2 of the Gaussian network into the precision matrix W of the equivalent Gaussian distribution is achieved by the following recursive formula [25]:

$$W(i+1) = \begin{pmatrix} W(i) + \frac{b_{i+1}b_{i+1}^t}{\sigma_{i+1}^2} & \frac{-b_{i+1}}{\sigma_{i+1}^2} \\ \frac{-b_{i+1}^t}{\sigma_{i+1}^2} & \frac{1}{\sigma_{i+1}^2}, \end{pmatrix} \tag{4}$$

where $W(i)$ is the $i \times i$ upper left submatrix of W, $W(1) = \frac{1}{\sigma_i^2}$, b_i is the column vector $(b_{1i}, ..., b_{(i-1)i})^t$ and b_i^t is the transposed vector.

– **Learning**

The Gaussian network learning from a dataset can be stated as follows. Given a collection of observed data, find the network model to explain these data with maximum likelihood. By finding the network we mean to provide the structure of the graph, as well as the probability distribution of each variable that best fits the data.

One usual approach to this task is to adopt a procedure for searching the space of all possible candidate network structures, given a metric that can provide a relative score to each point in the space. The heuristic search utilized by GAIS begins with an empty network, i.e. with no edges. Next, the probability distribution of each variable is estimated using the dataset and the score of the network is computed. The search process proposes small changes to the structure in order to obtain a network with higher score than the previous one. These small changes can be accomplished by adding or deleting an edge, or reversing the edge direction. Every time a change is made it is necessary to compute the probability distribution of the variables for the modified network. Regarding the scoring metrics, GAIS utilizes the *Bayesian Information Criterion* (BIC) for Gaussian networks [24]:

$$p(D|G) = \left[\prod_{l=1}^{N} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi v_i}} e^{-1/2v_i(x_{li} - m_i - \sum_{x_k \in \boldsymbol{pa_i}} b_{ki}(x_{lk} - m_k))^2} \right] - f(N) * dim(G), \tag{5}$$

where D represents the dataset, G is the network under evaluation, N is the number of instances, n is the number of variables, b_{ki} is the linear coefficient of relationship between X_k and X_i, $\boldsymbol{pa_i}$ is the set of parents of X_i and v_i is the conditional variance of X_i given $X_1, ..., X_{i-1}$ $\forall i, k$. The function $f(N) = \frac{1}{2}\, ln\, N$ is responsible for penalizing complex models and $dim(G) = 2n + \sum_{i=1}^{n} \boldsymbol{pa_i}$ is the number of parameters to be estimated.

– **Sampling**

Once the Gaussian network is built, we can generate new instances using the joint probability distribution encoded by the network (Eq. 1). To accomplish this task, we utilize a method that finds an ancestral ordering of the nodes in the Gaussian network and instantiates one variable at a time in a forward manner, that is, a variable is not sampled until all its parents have already been sampled [26]. In GAIS, the number of sampled instances varies with the problem.

3.2 GAIS with a Gaussian Mixture Model

GAIS supposes that the available dataset for building the probabilistic model was generated by a single multivariate Gaussian distribution. To solve unimodal functions, the algorithm achieves a good estimation of the probability distribution. However, to solve multimodal functions, where the local optima are not concentrated in only one region, but scattered, a single multivariate Gaussian distribution can not model the data satisfactorily.

Therefore, we have also developed a version of GAIS that utilizes a Gaussian mixture model, namely GAIS$_M$. The components of the mixture are defined by means of clustering. The algorithm groups the selected solutions and process each group separately. This means that, for each group, a Gaussian network is generated using the same procedure as before. GAIS applies k-means to cluster the solutions, due to its simplicity and effectiveness. The number of clusters, k is specified empirically.

With this modification, the density function of joint probability of the best solutions is said to be a mixture of Gaussian probability density functions, expressed by:

$$f(x) = \sum_{i=1}^{k} \alpha_i f_i(x), \quad \sum_{i=1}^{k} \alpha_i = 1, \quad \alpha_i \geq 0, \tag{6}$$

where $f_i(x)$ is a single multivariate Gaussian probability density function, k is the number of clusters and, consequently, the number of components of the mixture, α_i is the coefficient of the i-th component of the mixture. The value of each α_i is proportional to the number of elements in the i-th cluster:

$$\alpha_i = \frac{c_i}{\sum_{j=1}^{k} c_j}, \quad i = 1, ..., k, \tag{7}$$

where c_i is the number of elements in cluster i.

4 Experiments

This section describes the experiments carried out to evaluate the proposed algorithm. We have applied GAIS to eight well-known problems and compared the performance with other optimization tools reported in the literature.

4.1 Test Functions

Eight functions often utilized in the literature were tested during the experiments. In what follows, we provide a description of each function for a generalized dimensionality d.

- **Sphere:** This function is probably the most standard unimodal benchmark problem for optimization. It involves the minimization of a single hyperparabola and the minimum value for any dimensionality is 0 which is obtained if all x_i take the value of 0.

$$F_1(x) = \sum_{i=1}^{d} x_i^2, \quad x_i \in [-100; 100]^d. \tag{8}$$

- **Summation Cancellation (SumCan):** This problem has multivariate linear interactions between the variables. So, algorithms that are capable of modelling these dependencies are supposed to outperform algorithms that are not capable of doing so. The optimum is located at a very sharp peak, which implies that the optimization algorithm needs to be able to prevent premature convergence in order to reach the global optimum.

$$\begin{array}{l} F_2(x) = 100/(10^{-5} + \sum_{i=1}^{d} |y_i|), \quad x_i \in [-3; 3], \\ \text{where} \;\; y_1 = x_1, \;\; y_i = x_i + y_{i-1}, \;\; i \geq 2. \end{array} \tag{9}$$

- **Rosenbrock:** It is a highly nonlinear function. It has a curved valley along which the quality of the solutions is much better than in its neighborhood. This valley has a unique minimum for any dimensionality, which is obtained when all x_i are set to the value of 1. Rosenbrock's function has proven to be a real challenge for any algorithm.

$$F_3(x) = \sum_{i=1}^{d-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, \quad x_i \in [-5.12; 5.12]^d. \tag{10}$$

- **Griewank:** It is a function with many local optima. Basically, it is a parabola superimposed with a sine function to obtain these local optima. The minimum value for this function for any dimensionality is 0, which is obtained if all x_i are set to the value of 100.

$$F_4(x) = 1 + \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}}), \quad x_i \in [-600; 600]^d. \tag{11}$$

– **Ackley:** The value of the global optimum is 0 for any dimensionality, which is obtained when all x_i take the value of 0.

$$F_5(x) = -20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)} + 20 + e, \quad x_i \in [-32.768; 32.768]^d. \tag{12}$$

– **Michalewicz:** It is a function with many local optima. Michalewicz's function has many long channels along which the minimum value throughout the channel is the same. The minimum value for this function depends on its dimensionality.

$$F_6(x) = -\sum_{i=1}^{d} \sin(x_i)\sin^{20}\left(\frac{(i+1)x_i^2}{\pi}\right), \quad x_i \in [0; \pi]^d. \tag{13}$$

– **Rastrigin:** Rastrigin's function has many local minima, but just one global optimum. The value of global optimum is 0 for any dimensionality, which is obtained when all x_i are set to 0.

$$F_7(x) = 10d\sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i)), \quad x_i \in [-5.12; 5.12]^d. \tag{14}$$

– **Schwefel:** It is a function with many peaks and valleys. The global minimum is located far from local minima, leading the algorithms to get stuck at these local minima. The value of the global optimum is -418.9829d, which is obtained when all variables take the value 420.9687.

$$F_8(x) = \sum_{i=1}^{d} -x_i \ \sin(\sqrt{|x_i|}), \quad x_i \in [-500; 500]^d. \tag{15}$$

4.2 Experimental Setup

All functions were tested using 30 dimensions, except for Michalewicz's function where we adopted 10 dimensions. The reason to use a 10-dimensions Michalewicz's function is that in the literature this function has been considered in this way. This is the same reason why we adopted 30 dimensions for the remaining cases.

The parameters of GAIS and GAIS$_M$ are the same for all cases. The initial population contains 100 antibodies. The number of components of the mixture for GAIS$_M$ was established empirically as 3.

We have compared the proposed algorithms with two alternative approaches. The first one is the Iterated Density Estimation Algorithm (IDEA) [27], that also utilizes a Gaussian mixture model. IDEA applies the *Leader* clustering algorithm [28] to generate the components of the mixture. The number of clusters was also defined as 3 and the population size was equal to 500. The other algorithm considered for comparative analysis is the Artificial Immune Network for Optimization (opt-aiNet) [29]. The initial population contains 100 antibodies and each one generates 2 clones. Parameters of both, opt-aiNet and IDEA, were adjusted following their authors' guidelines.

Notice that opt-aiNEt is an immune-inspired algorithm not endowed with probabilistic models, and IDEA uses probabilistic models but are not endowed with the search capabilities of immune-inspired algorithms. The authors believe that using these two algorithms as contenders is justifiable in an inaugural paper proposing a Gaussian immune-inspired algorithm, because it allows two reasonable contrasts: (*i*) immune-inspired against non-immune-inspired Gaussian algorithms; (*ii*) Gaussian against non-Gaussian immune-inspired algorithms.

For GAIS, GAIS$_M$ and IDEA, the Gaussian network is built at every 10 iterations, but the probabilities of the network are updated at every iteration. In order to penalize the complexity of the model, we have imposed a constraint on the number of parents a node can have. It corresponds to a maximal order of interactions that can be covered and it directly influences the complexity of the model. By our previous experience on Gaussian network learning, we know that when the complexity of the network is too high, it is more likely to detect spurious correlations on the data. Thus, each variable can have only two parents. For GAIS and GAIS$_M$, 80% of the best solutions are utilized to build the probabilistic model the number of samples generated is half of the size of the current population. The number of random individuals inserted into the population in order to create diversity is around 3% of the current population size. The stopping condition for all algorithms is the maximum number of function evaluations, defined as 10^5 for all benchmarks.

4.3 Results

The average results obtained by the four algorithms over 30 executions are presented in Table 1. To evaluate the statistical significance of the difference among the performance of the algorithms, the *t-test* was employed with a significance level of 95% (α=0.05). The symbol ⋇ denotes there is a statistical difference between the results obtained by GAIS and the compared algorithm. On the other hand, the symbol † denotes a statistical difference related to GAIS$_M$.

Observing Table 1, we can see that all algorithms have found the global optimum for Sphere function. In the case of Summation Cancellation, which is a function with strong interdependencies among the variables, GAIS has found the global optimum whereas GAIS$_M$ and IDEA have achieved a value very close to the global optimum. This indicates that these algorithms really have captured the relationship between the variables. On the other hand, opt-aiNet performs very badly in this function because it is not provided with a similar mechanism. For the remaining functions, our algorithms achieved better results, with a slight advantage to GAIS$_M$. Once GAIS$_M$ utilizes a mixture model, generally it can fit the data (promising solutions) more satisfactorily than the other algorithms. Although IDEA also utilizes a mixture model, its mechanism to exploit/explore the search space is not so good as in GAIS$_M$, leading the algorithm to get stuck at local minima easily.

Table 1. Average results over 30 executions obtained by GAIS, GAIS$_M$, IDEA and opt-aiNet on the eight benchmark functions

Function	GAIS	GAIS$_M$	IDEA	opt-aiNet
F_1	**0** $\pm$ 0.000	**0** $\pm$ 0.000	**0** $\pm$ 0.000	**0** $\pm$ 0.000
F_2	$\mathbf{10^7}$ $\pm$ 0.0	9.87×10^6 $\pm$ 4169 (⁎)	7.44×10^6 $\pm$ 13996 (⁎†)	6.82×10^6 $\pm$ 9502 (⁎†)
F_3	**0.8417** $\pm$ 0.007	0.9025 $\pm$ 0.019	1.091 $\pm$ 0.056 (⁎†)	7.67 $\pm$ 1.245 (⁎†)
F_4	0.0058 $\pm$ 0.002	**0** $\pm$ 0.000 (⁎)	0.037 $\pm$ 0.022 (⁎†)	0.403 $\pm$ 0.066 (⁎†)
F_5	4.4×10^{-6} $\pm$ 0.003	**0** $\pm$ 0.000 (⁎)	0.0008 $\pm$ 0.007 (⁎†)	0.096 $\pm$ 0.002 (⁎†)
F_6	-9.6539 $\pm$ 0.016	**-9.6604** $\pm$ 0.027	-9.6257 $\pm$ 0.019 (⁎†)	-9.0841 $\pm$ 0.086 (⁎†)
F_7	0.0028 $\pm$ 0.001	**0.0007** $\pm$ 0.004	0.0032 $\pm$ 0.003 (†)	1.96 $\pm$ 0.907 (⁎†)
F_8	-12569.87 $\pm$ 0.048	**-12569.49** $\pm$ 0.001	-12569.32 $\pm$ 0.460	-8903.24 $\pm$ 0.734 (⁎†)

Table 2. Average number of fitness evaluations until the best value for the eight functions is reached

Function	GAIS	GAIS$_M$	IDEA	opt-aiNet
F_1	6719.2 $\pm$ 291.7	**6485.9 $\pm$ 314.2**	11038.6 $\pm$ 392.5	9076.8 $\pm$ 427.3
F_2	41658.8 $\pm$ 1326.4	**37014.0 $\pm$ 1244.9**	80255.3 $\pm$ 2072.1	143892.4 $\pm$ 1705.4
F_3	81037.3 $\pm$ 2188.3	**75368.2 $\pm$ 2046.3**	142604.8 $\pm$ 4192.7	92831.2 $\pm$ 3064.2
F_4	**7213.5 $\pm$ 1721.6**	8644.2 $\pm$ 1645.8	19016.6 $\pm$ 2031.5	27513.7 $\pm$ 2482.6
F_5	7508.1 $\pm$ 1840.7	**5542.5 $\pm$ 1112.3**	23864.2 $\pm$ 2970.4	15376.9 $\pm$ 2062.4
F_6	**3052.8 $\pm$ 1389.5**	4907.4 $\pm$ 1503.3	17046.1 $\pm$ 2192.7	13594.1 $\pm$ 2485.8
F_7	4782.2 $\pm$ 1762.9	**3381.6 $\pm$ 1040.8**	11406.9 $\pm$ 2826.4	36033.3 $\pm$ 3064.6
F_8	**5368.4 $\pm$ 1065.2**	6194.8 $\pm$ 1249.2	27324.3 $\pm$ 2240.6	19740.5 $\pm$ 1740.6

4.4 Discussion

As stated before, GAIS offers conceptual advantages over the contenders and they are described in what follows.

The first advantage is related to the maintenance of building blocks. With this mechanism, GAIS avoids disrupting the partial solutions found so far, leading to a great improvement in the quality of the candidate solutions even in the first iterations. We have observed during the experiments that while GAIS and GAIS$_M$ have found high-quality solutions with few fitness evaluations, the other methodologies needed more fitness evaluations to achieve a satisfactory performance. Table 2 shows the average number of fitness evaluations until the algorithms reach the best solution. Observing Table 2, we note that our proposals are able to find the best solution much faster than the other algorithms.

Other aspect to be highlighted is the effective mechanism of GAIS and GAIS$_M$ to perform a multimodal search, finding diverse high-quality local optima quickly, as depicted in Table 3. In addition, the initial population size is not crucial to GAIS due to its capability to control the population size along the search process in response to the particularities of the problem.

Regarding the implementation of GAIS and GAIS$_M$, we notice that the algorithms do not require a large amount of computational resources [30]. Although a Gaussian network has to be produced, the proposed methodology still preserves

Table 3. Average number of local optima found by the algorithms

Algorithm	F_3	F_4	F_5	F_6	F_7	F_8
GAIS	20.2 ± 4.1	40.3 ± 5.7	29.2 ± 5.3	21.7 ± 6.2	16.0 ± 3.4	19.5 ± 5.2
GAIS$_M$	22.7 ± 3.8	42.1 ± 6.3	33.7 ± 5.2	20.4 ± 4.7	18.6 ± 6.1	20.9 ± 3.2
IDEA	9.8 ± 3.3	16.8 ± 7.0	14.5 ± 4.6	11.7 ± 5.4	8.6 ± 2.7	10.8 ± 4.1
opt-aiNet	20.6 ± 3.7	36.4 ± 9.3	26.2 ± 3.4	18.5 ± 4.6	20.1 ± 5.8	17.3 ± 3.8

the computational tractability due to the restriction of at most two parents for each node in the network. Additionally, we avoid the synthesis of the network at each iteration.

5 Concluding Remarks and Future Work

In this paper we have proposed a novel immune-inspired algorithm for solving continuous optimization problems, taking into account the interdependencies between the variables of the problem. Our proposal, called Gaussian Artificial Immune System (GAIS), replaces the traditional mutation and cloning operators with a probabilistic model representing the joint distribution of promising solutions and, subsequently, utilizes this model for sampling new solutions. The probabilistic model used is a Gaussian network due to its capability to properly capture the most relevant interactions among the variables of the problem. Two versions of the algorithm were developed. The first one utilizes a single multivariate Gaussian probability distribution and the other one utilizes a Gaussian mixture model. In order to alleviate the computational cost required to learn the structure of the Gaussian network, this task is performed at a certain number of iteration while the probabilities associated with the structure are updated at each iteration.

To evaluate the algorithm, we have considered eight functions and compared the obtained results with those produced by other approaches. The experiments pointed favorably to our proposal for all tested functions.

We are currently investigating some aspects that can be further improved, such as alternative clustering algorithms to generate the components of the mixture and even other methods, such as *Expectation Maximization* (EM). We are also analyzing the performance of the algorithms in more challenging problems. Furthermore, comparisons with other evolutionary algorithms will be carried out. Another aspect to be considered is the extension of the proposals to handle multiobjective optimization problems in continuous domains.

References

1. Timmis, J., Hone, A., Stibor, T., Clark, E.: Theoretical advances in artificial immune systems. Theoretical Computer Science 403(1), 11–32 (2008)
2. de Castro, L.N., Timmis, J.: An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm. Springer, Heidelberg (2002)
3. Dasgupta, D. (ed.): Artificial Immune Systems and Their Applications. Springer, Heidelberg (1999)

4. de Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Trans. Evolutionary Computation 6(3), 239–251 (2002)
5. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1992)
6. Castro, P.A.D., Von Zuben, F.J.: BAIS: A Bayesian Artificial Immune System for the effective handling of building blocks. Information Sciences 179(10), 1426–1440 (2009)
7. Castro, P.A.D., Von Zuben, F.J.: Feature subset selection by means of a Bayesian artificial immune system. In: Proc. Eighth International Conference on Hybrid Intelligent Systems, pp. 561–566 (2008)
8. Castro, P.A.D., Von Zuben, F.J.: MOBAIS: A Bayesian artificial immune system for multi-objective optimization. In: Bentley, P.J., Lee, D., Jung, S. (eds.) ICARIS 2008. LNCS, vol. 5132, pp. 48–59. Springer, Heidelberg (2008)
9. Castro, P.A.D., Von Zuben, F.J.: Multi-objective Bayesian artificial immune system: Empirical evaluation and comparative analyses. Journal of Mathematical Modelling and Algorithms 1, 151–173 (2009)
10. Castro, P.A.D., Von Zuben, F.J.: Multi-objective feature selection using a Bayesian artificial immune system. Journal of Intelligent Computing and Cybernetics (2010) (in press)
11. Dasgupta, D.: Advances in Artificial Immune Systems. IEEE Computational Intelligence Magazine, 40–49 (2006)
12. Ada, G.L., Nossal, G.J.V.: The clonal selection theory. Scientific American 257(2), 50–57 (1987)
13. Jerne, N.K.: Towards a network theory of the immune system. Ann. Immunol. (Inst. Pasteur) 125C, 373–389 (1974)
14. Goldberg, D.E., Deb, K., Kargupta, H., Harik, G.: Rapid accurate optimization of difficult problems using fast messy genetic algorithms. In: Proc. of the Fifth Int. Conf. on Genetic Algorithms, pp. 56–64 (1993)
15. Goldberg, D.E., Korb, G., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems, 493–530 (1989)
16. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
17. Mühlenbein, H., Paass, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
18. Baluja, S., Davies, S.: Using optimal dependency-trees for combinational optimization. In: Proc. of the 14th Int. Conf. on Machine Learning, pp. 30–38 (1997)
19. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Comput. Optim. Appl. 21(1), 5–20 (2002)
20. Pelikan, M.: Probabilistic Model-Building Genetic Algorithms. Springer, Heidelberg (2005)
21. DelaOssa, L., Gámez, J.A., Mateo, J.L., Puerta, J.: Avoiding premature convergence in estimation of distribution algorithms. In: Proc. of the 11th Congress on Evolutionary Computation, pp. 455–462 (2009)
22. de Franca, F.O., Coelho, G.P., Von Zuben, F.J.: On the diversity mechanism of opt-aiNet: a comparative study with fitness sharing. In: Proc. of the IEEE Congress on Evolutionary Computation (2010)
23. Geiger, D., Heckerman, D.: Learning Gaussian Networks. Technical Report MSR-TR-94-10, Microsoft Research (1994)

24. Lu, Q., Yao, X.: Clustering and learning Gaussian distribution for continuous optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C 35(2), 195–204 (2005)
25. Shachter, R.D., Kenley, C.R.: Gaussian influence diagrams. Management Science 35(5), 527–550 (1989)
26. Ripley, B.D.: Stochastic simulation. John Wiley & Sons, Chichester (1987)
27. Bosman, P., Thierens, D.: Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 767–776. Springer, Heidelberg (2000)
28. Hartigan, J.A.: Clustering algorithms. Wiley, New York (1975)
29. de Castro, L.N., Timmis, J.: An artificial immune network for multimodal function optimization. In: Proc. of the 2002 Congress on Evolutionary Computation, pp. 699–704 (2002)
30. Broom, B., Subramanian, D.: Computational methods for learning Bayesian networks from high-throughput biological data. In: Bayesian inference for gene expression and proteomics. Cambridge University Press, Cambridge (2006)