10

Surrogate-Assisted Artificial Immune Systems for Expensive Optimization Problems

Heder S. Bernardino, Leonardo G. Fonseca, and Helio J. C. Barbosa
Laboratório Nacional de Computação Científica, LNCC/MCT
Brazil

1. Introduction

When an animal is exposed to antigens an efficient immune response is developed in order to defend the organism where specific antibodies are produced to combat them. The best antibodies multiply (cloning) and are improved (hypermutation and replacement) while new antibodies, produced by the bone marrow, are generated. Thus, if the organism is again attacked by the same antigen a quicker immune response takes place. This scheme of adaptation is known as clonal selection and affinity maturation by hypermutation or, more simply, clonal selection (Garrett, 2004). Computational methods inspired by the biological immune system are called Artificial Immune Systems (AISs). Immune-inspired algorithms have found applications in many domains. One of the most important area, the optimization, is a mathematical principle largely applied to design and operational problems in all types of engineering, as well as a tool for formulating and solving inverse problems such as parameter identification in scientific and engineering situations. When applied to optimization problems, the AISs are stochastic populational search methods which do not require a continuous, differentiable, or explicit objective function, and do not get easily trapped in local optima.

However, the AISs, as well as other nature-inspired techniques, usually require a large number of objective function evaluations in order to reach a satisfactory solution. As modern problems have lead to the development of increasingly complex and computationally expensive simulation models, this becomes a serious drawback to their application in several areas such as Computational Structural Mechanics, Reservoir Simulation, Environmental Modeling, and Molecular Dynamics. Thus, a good compromise between the number of calls to the expensive simulation model and the quality of the final solutions must often be established.

A solution to this problem is to modify the search process in order to obtain either a reduction on the total computational cost or an increase in the efficiency of the optimization procedure. The solution considered here is the use of a surrogate model (or metamodel), which provides an approximation of the objective function, replacing the computationally intensive original simulator evaluation. Additionally, the surrogate model can help to smooth out the objective function landscape, and facilitate the optimization process.

The idea of reducing the computation time or improving the solutions performing less computationally expensive function evaluations can be found in the evolutionary computation literature (Bull, 1999; El-Beltagy et al., 1999; Jin, 2002; Zhou, 2004; Rasheed,

Source: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos,
ISBN 978-953-307-008-7, pp. 572, October 2009, I-Tech, Vienna, Austria

2005; Forrester, 2009). Exist many surrogate models available: some examples are polynomial models (Response Surface Methodology) (Grefenstette & Fitzpatrick, 1985), Artificial Neural Networks (Ferrari & Stengel, 2005), Kriging or Gaussian Processes (Kecman, 2001), Radial Basis Functions (Giannakoglou, 2002; Forrester, 2009), and Support Vector Machines (Emmerich et al., 2006). In addition, several surrogates may be derived from physical or numerical simplifications of the original simulation model.
In this paper we propose an artificial immune system assisted by a Similarity-Based Surrogate Model (SBSM) in which the objective is to allow the AIS to evolve for a larger number of generations, but still using a fixed number of expensive evaluations, in order to obtain improved final solutions.
This chapter is organized as follows. Section 2 gives a formulation for the optimization problems considered here. AISs are presented in Section 3. Sections 4 and 5 present the Surrogate Models and the surrogate-assisted AIS, respectively. The computational experiments and a discussion of the results obtained can be found in Section 6. The concluding remarks are given in Section 7.

2. The optimization problem

The class of optimization problems considered here can be written as

$$\begin{aligned} &\textit{Minimize} \quad f(\vec{x}) \\ &\textit{subject to} \\ &x_i^L \le x_i \le x_i^U, \quad i = 1, \cdots, n \end{aligned}$$

where $f(\vec{x})$ is the objective function to be optimized (it is easy to see that a maximization problem can also be solved by minimizing $-f(\vec{x})$), n is the number of design/decision variables, and the search space is bounded by the constraints $x_i^L \le x_i \le x_i^U, i = 1, \cdots, n$.
In practice, the value of $f(\vec{x})$ is normally computed by means of a simulator. Thus, the evaluation of a candidate solution is often computationally expensive. In the proposed algorithm, the individuals evaluated by the original function (i.e., solutions evaluated exactly) are stored in a database (memory cells). The population of memory cells is used to construct a surrogate, based on similarity, which is used along the optimization procedure to perform extra (surrogate) evaluations, resulting in a larger number of total (surrogate plus exact) evaluations. Those extra surrogate evaluations involve a simple procedure, with relatively negligible computational cost.

3. Artificial immune systems

AISs are computational techniques, inspired by the biological immune system, which can be used to solve complex real world problems. In optimization problems (Bernardino & Barbosa, 2009), the AIS algorithms evolve improved solutions by means of natural immune mechanisms, such as clonal selection, immune network theory, vaccination, or other immune system concepts.
In general, an immune optimization algorithm will have a population of antibodies (candidate solutions) and another composed by the antigens (objectives) that the antibodies attempt to reach or match (optimize). The main differences among the AIS techniques

applied to optimization problems reside in which natural immune mechanism is considered to evolve the antibodies, i.e., how the candidate solutions evolve.
According to clonal selection theory – the immune mechanism used by the algorithm considered here – there is a selection process which leads to the evolution of the immune system repertoire during the lifetime of the individual (Burnet, 1959). Also, according to this theory, on binding with a suitable antigen, activation of lymphocytes occurs. The clonal expansion is the process whereby clones of the activated lymphocyte are produced expressing receptors identical to the original one that encountered the antigen. Any lymphocyte that has receptors specific to molecules of its own body must be deleted (i.e., these lymphocytes will not be cloned). Therefore, only an antigen may cause a clonal expansion. Then, the clonal selection culminates in the increase in the average affinity between the antibodies and antigens due to the somatic hypermutation and selection mechanisms of clonal expansion. It is responsible for the fact that upon a subsequent exposure to the antigen, a stronger immune response is produced (AISWeb, 2009).
The clonal selection process is directly responsible for the evolution of the candidate solutions. The affinity maturation, as it is also known, is a mutation of the individuals applied with a high rate, which is inversely proportional to the fitness of the antibody (affinity antibody-antigen), unlike the standard mutation of Evolutionary Algorithms (EAs). Thus, inferior individuals are subject to more modification than the better ones, which need a finer tuning. When applied alone, this procedure is a random search. Therefore, a selection method is necessary to keep the good solutions, eliminate the worst ones, and maintain diversity.

3.1 Clonal selection algorithm

Based on the clonal selection theory, de Castro and Von Zuben proposed an AIS algorithm that performs computational optimization and pattern recognition tasks. CLONALG, or CSA (as it was initially called), evolves the antibodies inspired by the concept of clonal selection. In this method, each antibody is cloned, hypermutated (mutation applied with high rate), and those with higher affinity are selected. The main features of this technique are (i) the mutation rate, normally inversely proportional to the affinity of the antibody with respect to the antigens and (ii) the absence of recombination operators (such as crossover in GAs). The clonal selection principle can be interpreted as a remarkable microcosm of Darwinian evolution (Cziko, 1995) and can be considered an evolutionary algorithm.
In (de Castro & Zuben, 2000) the CSA was proposed as "a powerful computational implementation of the clonal selection principle" and applied to two optimization (multimodal optimization, and a 30-city instance of the Traveling Salesman Problem) and one pattern recognition problems (binary character recognition) showing its potential as a meta-heuristic to solve multimodal and combinatorial optimization problems.
The improved CSA is known as CLONALG and was proposed in (de Castro & Zuben, 2002). Benchmark problems were considered in order to evaluate the performance of the algorithm as well as a sensitivity analysis with respect to the user-defined parameters was presented.
Figure 1 shows CLONALG's pseudo-code which is inspired in the algorithm presented in (Bernardino & Barbosa, 2009).
In the algorithm of Figure 1, *antibodies* is a population of candidate solutions, β defines the number of clones generated by each antibody (it can be the same for all antibodies or proportional to their affinities), ρ is a parameter used to define the mutation rate (de Castro

& Zuben, 2002), *nSelection* is the number of the best antibodies selected to be cloned, and *bestAffinity* is the best value found by the AIS. Also, in the same algorithm, the following functions must be considered: "calcAffinities" calculates the affinities between each antibody and all antigens (in optimization problems this value often corresponds to the value calculated by the objective function); "select" selects the *nSelection* best individuals to be cloned; "clone" clones the selected antibodies; "hypermutate" applies the somatic hypermutation in generated clones; "update" replaces some antibodies by other ones from hypermutated clones; "stopCondition" verify if the stop condition is satisfied; and "getBestAffinity" returns the best solution found.

```
1  begin
2      affinities ⟵ calcAffinities(antibodies);
3      while not stopCondition() do
4          selectedAntibodies ⟵ select(antibodies, affinities, nSelection);
5          clones ⟵ clone(selectedAntibodies, affinities, β);
6          clones ⟵ hypermutate(clones, affinities, ρ);
7          cloneAffinities ⟵ calcAffinities(clones);
8          update(antibodies, affinities, clones, cloneAffinities);
9      bestAffinity ⟵ getBestAffinity(antibodies);
10 end
```

Fig. 1. A CLONALG pseudo-code for optimization problems.

The "update" method used here selects the best candidate solutions in the union of the antibody population and the newly generated set of clones. The idea is to use a replacement method as simple as possible, considering that the focus of this work is the use of the surrogate model. A pseudo-code for the "update" procedure can be found in Figure 2.

```
1  begin
2      for i = 0; i < λ; i ← i + 1 do
3          add(antibodies[i], clones);
4          add(affinities[i], cloneAffinities);
5      sort(clones, affinities);
6      i ⟵ 0;
7      while i < λ do
8          antibodies[i] ⟵ clones[i];
9          affinities[i] ⟵ cloneAffinities[i];
10         i ⟵ i + 1;
11 end
```

Fig. 2. Pseudo-code for "update" from Figure 1.

More information about artificial immune algorithms for optimization problems can be found in (Bernardino & Barbosa, 2009).

4. Surrogate models

Surrogate modeling, or meta-modeling, can be viewed as the process replacing the original evaluation function (a complex computer simulation) by a substantially less expensive approximation. The surrogate model should be simple, general, and keep the number of

control parameters as small as possible (Blanning, 1974). Similarity-Based Surrogate Models (SBSMs), an example of such surrogates, will be described in the following sections.

4.1 Similarity-Based Surrogate Models (SBSMs)

In contrast to "eager" learning algorithms such as Neural Networks, Polynomial Response Surfaces, and Support Vector Machines, which generate a model and then discard the inputs, the Similarity-Based Surrogate Models (SBSMs) store their inputs and defer processing until a prediction of the fitness value of a new candidate solution is requested. Thus, SBSMs can be classified as "lazy" learners or memory-based learners (Aha, 1997) because they generate the output value by combining their stored data using a similarity measure. Any intermediate structure or result is then discarded.

Fitness Inheritance, Fitness Imitation, and the nearest neighbors method can be classified as SBSMs. The following sections present these approaches and describe in detail the nearest neighbor method, which is the surrogate model used here.

4.1.1 Fitness inheritance

First proposed in (Smith et al., 1995), the fitness inheritance surrogate model has been applied in several problems (Bui et al., 2005; Ducheyne et al., 2003; 2007; Salami & Hendtlass, 2003; Sastry et al., 2004; Zheng et al., 1997) and algorithms (Pilato et al., 2008; Reyes-Sierra & Coello, 2005). In this method, all the individuals in the initial population have their fitness value calculated by the exact objective function evaluator. Thereafter, a fraction of the individuals in the population has its affinity (or fitness) values inherited from their parents, while the remaining candidate solutions are evaluated using the original (exact) objective function.

Given a candidate solution x_h generated from the parents x_{p_i}, with $i = 1,2$, the surrogate evaluation is given by:

$$\hat{f}(x_h) = \begin{cases} f(x_{p_i}) & \text{if } s(x_h, x_{p_i}) = 1,\ i = 1,2 \\ \dfrac{s(x_{p_1}, x_h) f(x_{p_1}) + s(x_{p2}, x_h) f(x_{p_2})}{s(x_{p_1}, x_h) + s(x_{p2}, x_h)} & \text{otherwise} \end{cases}$$

where $s(x_{pi}, x_h)$ is the similarity between x_{pi} and x_h.

This kind of surrogate model assumes that the offspring are similar to their parents and thus their fitness values are determined as the weighted average of the parent's fitness. Although this approach introduces some noise in the search process and may adversely affect the final solution found (Ducheyne et al., 2007), it can be orders of magnitude less expensive than the original fitness evaluation. In the inheritance procedure an entire simulation is replaced by a technique with negligible computational cost, which may lead to large computational savings which grow with the rate of application of the inheritance technique and the cost of the fitness function evaluation (Chen et al., 2002; Sastry et al., 2001).

4.1.2 Fitness imitation

In the fitness imitation surrogate model (Jin, 2005) the individuals are clustered into groups. This task can be performed by any clustering technique (Kim & Cho, 2001). Each cluster can be represented by a candidate solution. The choice of the representative individual can be

made either deterministically or randomly (Mota & Gomide, 2006). The representative individuals are evaluated exactly while the other individuals in the same cluster will be approximated by the value of the representative solution and a similarity measure. When a new individual does not belong to any existing cluster it is evaluated by the original function. The term Fitness Imitation is used in contrast to Fitness Inheritance.
Figure 3 shows an illustration of Fitness Imitation, where the clusters are represented by dotted circles. The candidate solutions inside the same dotted circles belong to the same cluster. Black squares denote the representative individuals, i.e., those evaluated by the exact function. The remaining individuals (black circles) are evaluated by the surrogate model.

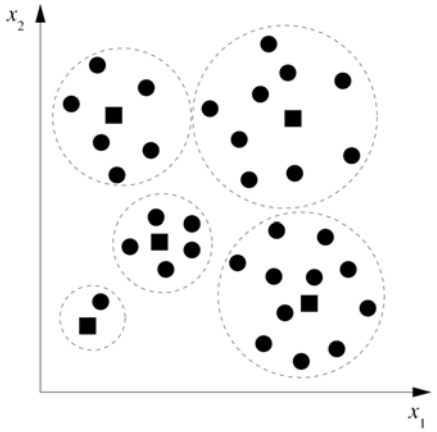


Fig. 3. Illustration of the Fitness Imitation organization.

4.1.3 Nearest neighbors

The nearest neighbors ($k\mu - NN$) is a surrogate model where the fitness values calculated are based on a set of samples μ, where $|\mu| = \eta$, evaluated exactly. Given a candidate solution x_h then we have that

$$\hat{f}(x_h) = \begin{cases} f(x_{\mu_i}) & \text{if } s(x_h, x_{\mu_i}), i = 1, \cdots, \eta \\ \dfrac{\sum_{i=1}^{k_\mu} s(x_h, x_{\mu_i})^u f(x_{\mu_i})}{\sum_{i=1}^{k_\mu} s(x_h, x_{\mu_i})^u} & \text{otherwise} \end{cases}$$

where $s(x_h, x_{\mu_i})$ is a similarity measure between x_h and $x_{\mu_i} \in \mu$ of the k_μ candidate solutions most similar to x_h, and u is set to 2.
For the binary-coded AIS, two similarity measures can be used. The first one, based on the Hamming distance, is given by

$$s(x_h, x_i) = 1 - \frac{d_H(x_h, x_i)}{l_c}$$

while the similarity measure based on the Euclidean distance is written as

$$s(x_h, x_i) = 1 - \frac{d_E(x_h, x_i)}{d_E(x^L, x^U)}$$

where $d_H(x_h, x_i)$ and $d_E(x_h, x_i)$ are respectively the Hamming and Euclidean distances between x_h and x_i, and l_c is the chromosome length.

The $k\mu - NN$ technique has several advantages: it is general, does not require any predefined functional form nor rely on any probability distribution, the variables can be either continuous or discrete, the databases are easy to maintain and can be updated when it is necessary to add or remove candidate solutions.

Although there is no training procedure associated, the computational cost for evaluating an individual is $O(k_\mu \eta)$ because of the search for the nearest neighbors.

5. Surrogate-assisted artificial immune system

Due to its simplicity, the Nearest Neighbors technique has been chosen to be used as the surrogate model. Although the Fitness Inheritance surrogate model is simpler than the Nearest Neighbor surrogate, it cannot be directly applied to the AIS algorithm. In the AIS, offspring are generated by cloning and hypermutation, and hence the fitness value of only one parent is available, while Fitness Inheritance requires at least two parents in order to build a surrogate evaluation.

Once a surrogate model has been chosen, there are many ways of introducing it into the original algorithm. Several approaches have been made in general surrogate-assisted evolutionary frameworks such as: integrating GAs with surrogate approximations (Queipo et al, 2005; Regis & Shoemaker, 2004) or landscape approximations (Knowles, 2006), the use of surrogate-guided evolutionary operators (Rasheed, 2002), surrogate-assisted local search (Lim et al, 2008; Wanner et al. 2008), accelerating the optimization process using surrogate models, pre-selection approaches (Giannakoglou, 2002; Praveen & Duvigneau, 2009), multiple surrogates (Acar & Rais-Rohani, 2008; Lim et al, 2008; Sanchez et al. 2007), and co-evolution of fitness predictors (Schmidt & Lipson, 2008). However, no Surrogate-Assisted AIS algorithm seems to have been proposed in the literature so far.

In this chapter we introduce the surrogate models into the immune inspired algorithm cycle by means of a model management procedure which, in each iteration, uses in a cooperative way both surrogate and exact models.

The first model management used here will be referred to as Random Selection (RS), i.e., a candidate solution is evaluated by the exact function, with probability $0 \le p_{sm} \le 1$. Therefore, evaluation by the surrogate model will occur with probability $1 - p_{sm}$. It is important to notice that RS applies to all clones except the ones from the best candidate solution which are evaluated exactly. This is due to the fact that the Nearest Neighbors surrogate model (Section 4.1.3) never generates a value better than the best neighbor. As a result, $\hat{\lambda} \le (\lambda - 1)\beta$ new antibodies are chosen at random to be evaluated by the surrogate model while $\lambda\beta - \hat{\lambda}$ candidate solutions are evaluated by the exact objective function, where λ is the population size. It is easy to see that $p_{sm} = 1 \Rightarrow \hat{\lambda} = 0$ and the standard CLONALG is recovered. However, $p_{sm} = 0$ does not mean that all evaluations will be performed by the surrogate model. In fact, in this case, only the clones from the best

antibody will be evaluated exactly. The modification is confined to the procedure "calcAffinities" from pseudo-code presented in Figure 1, where the decision is made as to using the surrogate model or not. A pseudo-code for the "calcAffinities" procedure can be found in Figure 4.

```
1  begin
2      i ⟵ 0;
3      foreach clone in clones do
4          if random() < p_sm or i < β then
5              cloneAffinities[i] ⟵ simulator(clone);
6              add(clone, memoryCells);
7              add(cloneAffinities[i], memoryCellAffinities);
8          else
9              cloneAffinities[i] ⟵ surrogateModel(memoryCells, cloneAffinities[i], clone);
10         i ⟵ i + 1;
11 end
```

Fig. 4. Pseudo-code for "calcAffinities" from Figure 1 – Random Selection (RS) model management.

It is important to notice that the initial population of candidate solutions is evaluated exactly. Also, every individual evaluated by the exact function is stored to be used by the surrogate model. In the immunological paradigm, this database of antibodies corresponds to the memory-cells set: a sample of representative cells stored with the objective of improving the combat against the antigens in the subsequent attacks.
The Surrogate-Assisted AIS developed here will be referred to as SAAIS.

6. Computational experiments

The impact of the introduction of the surrogate model into the CLONALG algorithm is analyzed in this section. The original algorithm and the proposed one (using a surrogate model) are evaluated by means of a set of benchmark unconstrained minimization problems from the literature. The performance comparison is made varying the value of the parameter p_{sm} from 1 (original algorithm) down to 0.1 (in steps of 0.1). As p_{sm} decreases, more surrogate evaluations are introduced into the evolutionary optimization process. In both cases the genotypical (Hamming) as well as the phenotypical (Euclidean) similarity measures are analyzed. Except for the use of the surrogate model, the algorithms are compared under the same set of parameters. The algorithmic parameters of the SBSM-CLONALG are summarized in Table 1.
Table 2 shows the set of 8 benchmark unconstrained minimization problems, with the respective name, explicit representation, maximum number of exact evaluations (simulations, N_{fMax}), and lower and upper bounds ($\left[x^L; x^U\right]$). These functions were chosen because they are commonly used in the literature and have different features (such as long narrow valleys, discontinuities, noise, and a large number of significant local optima). In all cases, the dimension considered is $n = 10$ and the optimal objective function value is $f^* = 0$.

Parameters	Value
Population size (λ)	30
Representation	Binary Gray Code with 20 bits
ρ (used by hypermutation)	4
β (number of clones)	1, 2, and 3
k_μ (number of neighbors)	2 and 4
Stop criterion	Maximum number of exact evaluations (N_{fMax})
Number of independent runs	50
New individuals randomly generated	0

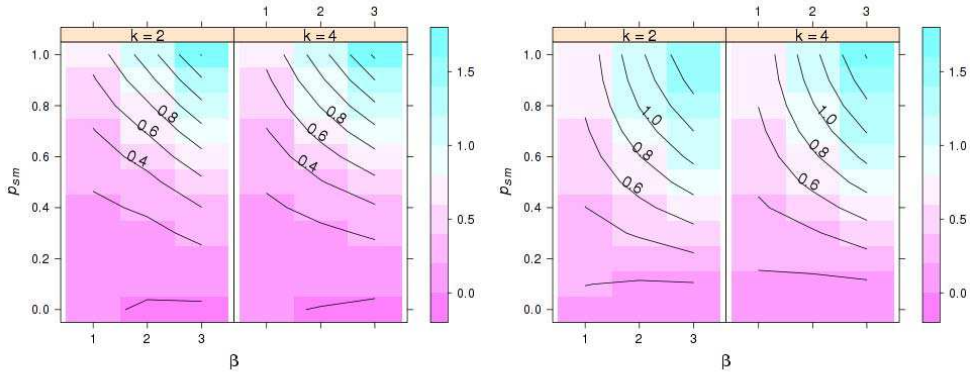
Table 1. Parameters used in all computation experiments

#	Name	Explicitly Function	N_{fMax}	$[x^L; x^U]$
F_{01}	Sphere	$\sum_{i=1}^{n} x_i^2$	2000	$[-5.12;5.12]$
F_{02}	Step	$\sum_{i=1}^{n} (x_i + 0.5)^2$	1000	$[-100;100]$
F_{03}	Griewank	$1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \frac{\cos(x_i)}{\sqrt{i}}$	3000	$[-600;600]$
F_{04}	Ackley	$20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}} - e^{\frac{\sum_{i=1}^{n} \cos(2\pi x_i)}{n}}$	3000	$[-32.768;32.768]$
F_{05}	Rosenbrock	$\sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right]$	10000	$[-5.12;5.12]$
F_{06}	Quarticnoise	$\sum_{i=1}^{n} i x_i^4 + U(0,1)$	4000	$[-4.28;4.28]$
F_{07}	Rastrigin	$\sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$	6000	$[-5.12;5.12]$
F_{08}	Schwefel	$418.982887272433n - \sum_{i=1}^{n} \left[x_i \sin\left(\sqrt{\lvert x_i \rvert} \right) \right]$	12000	$[-500;500]$

Table 2. Unconstrained minimization problems considered in the experiments

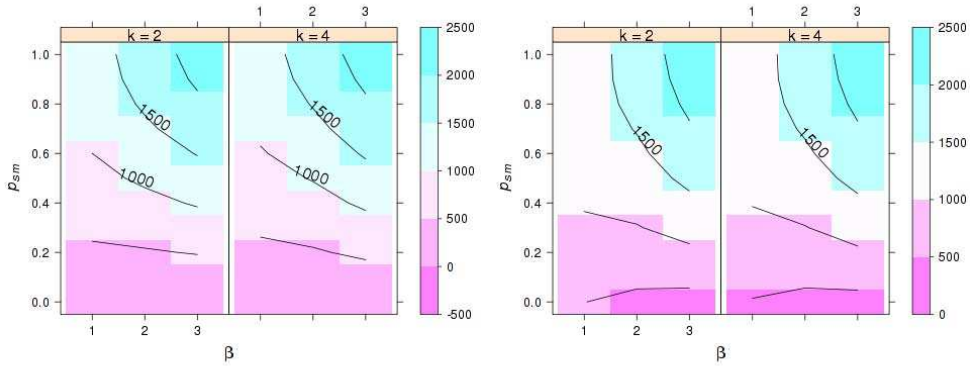
6.1 Random selection model management

In this section we analyze the impact of the parameters of the surrogate model (number of neighbors k_μ) and the algorithm (number of clones β) on the final results obtained by the SAAIS. Figures 7-14 show the contour plots of the fitness (averaged in 50 runs) obtained for functions F_{01} - F_{08} by the SAAIS, for different values of p_{sm} and number of clones. Each figure displays the results corresponding to the use of 2 and 4 neighbors to construct the surrogate model. The results for (a) Hamming similarity and (b) Euclidean similarity are shown in the figures.



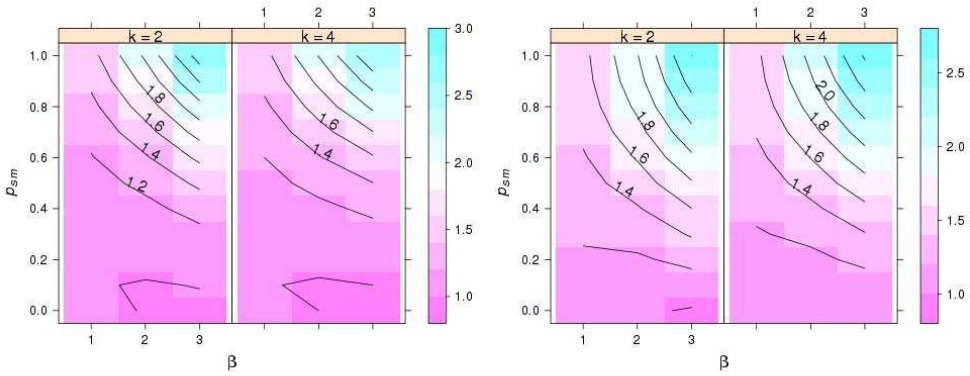
(a) (b)

Fig. 7. F_{01} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.



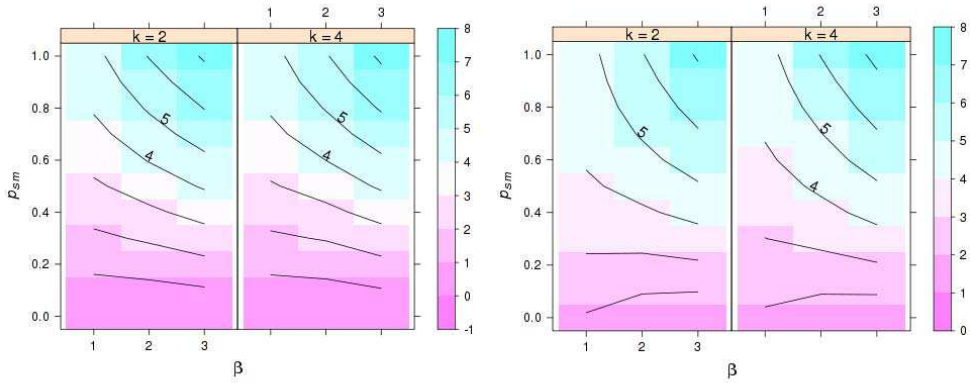
(a) (b)

Fig. 8. F_{02} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.



(a) (b)

Fig. 9. F_{03} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.



(a) (b)

Fig. 10. F_{04} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.

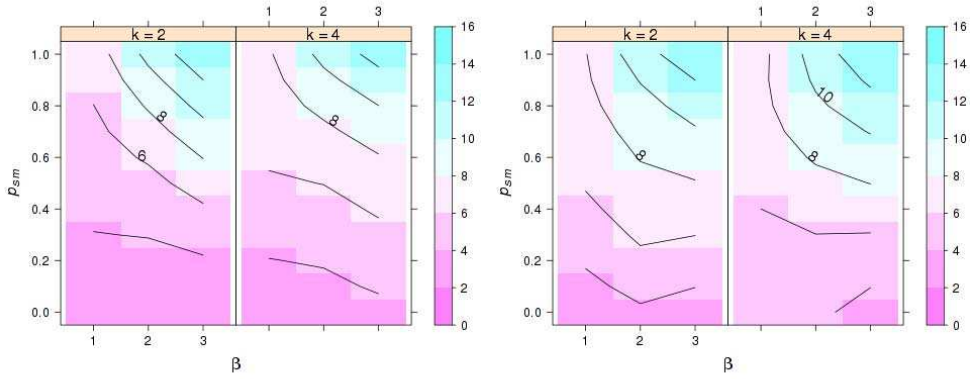


Fig. 11. F_{05} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.

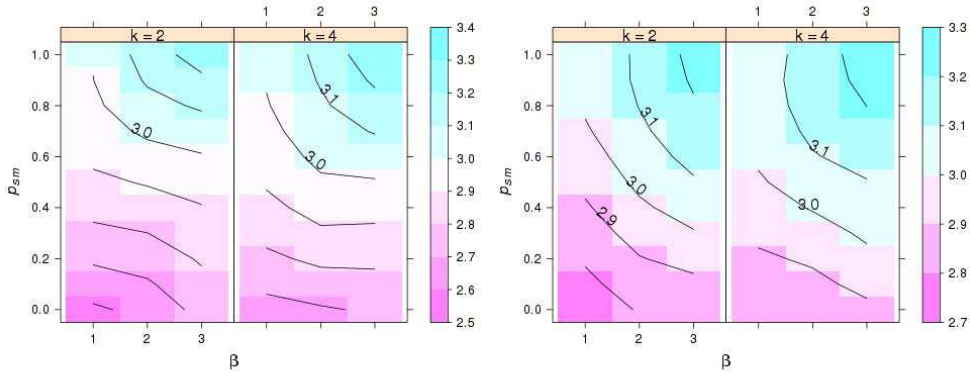
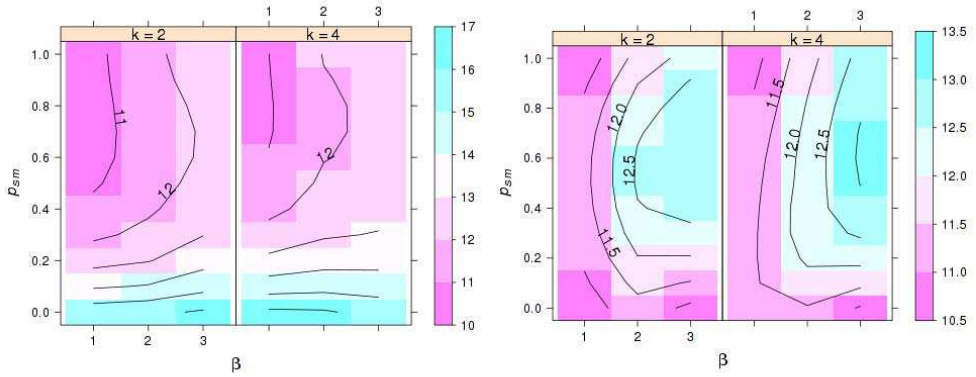
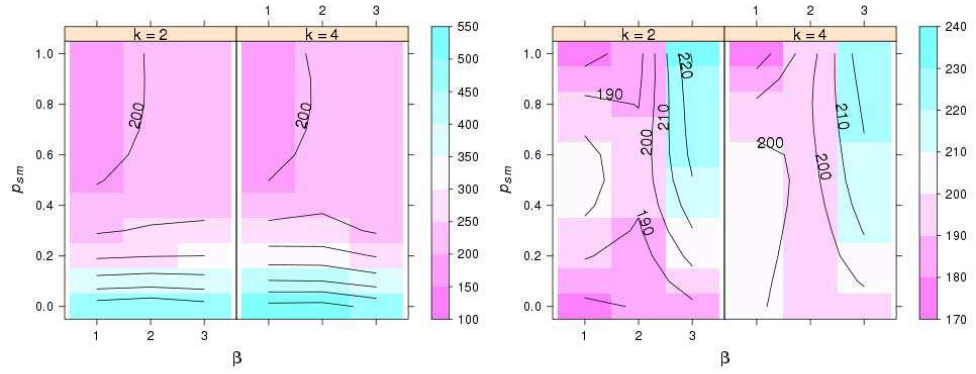


Fig. 12. F_{06} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.



(a) (b)

Fig. 13. F_{07} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.



(a) (b)

Fig. 14. F_{08} - Contour plots of the fitness (averaged in 50 runs) for different number of clones (1, 2 and 3), using 2 and 4 neighbors to build the surrogate. Results for: (a) Hamming similarity, and (b) Euclidean similarity.

In order to check whether there is a linear relationship between the algorithm performance (fitness) and its parameters (p_{sm}, number of neighbors and clones, and similarity measure) an analysis of correlation was performed.

For each combination of the different levels of each factor 50 independent runs were performed. As we have 2 levels for the similarity measures (Hamming and Euclidian), 3 levels for the number of clones (1, 2, and 3), 2 levels for the number of neighbors (2 and 4) and 10 levels for the parameter p_{sm} (1-0 varying in steps of 0.1), 50x2x3x2x10 = 6000 runs were performed for each test problem. Table 3 shows the correlations found among the dependent factor (averaged fitness values) and the four independent factors (p_{sm}, number of clones, similarity measure and number of neighbors). In order to obtain the results presented in that table, the categorical values for the Euclidean and Hamming similarity measures were set to 0 and 1, respectively. The Pearson's correlation coefficient can take values between -1 (perfect negative linear correlation) and 1 (perfect positive linear correlation).

From Table 3 we can see that the number of neighbors is weakly correlated to the final fitness, and thus using 2 or 4 neighbors to build the surrogates does not affect in a significant way the fitness values in the final population of the SAAIS. Also, we can observe the same weak correlation between the similarity measure (Euclidean or Hamming) and the fitness values.

Function	Fitness	p_{sm}	Clones	Similarity	Neighbors
F_{01}	1.000	0.811	0.422	0.227	-0.004
F_{02}	1.000	0.828	0.399	0.215	0.001
F_{03}	1.000	0.786	0.436	0.195	-0.014
F_{04}	1.000	0.876	0.323	0.227	-0.011
F_{05}	1.000	0.755	0.322	0.186	0.056
F_{06}	1.000	0.787	0.349	0.268	0.119
F_{07}	1.000	-0.265	0.593	-0.114	0.112
F_{08}	1.000	-0.403	0.175	-0.410	0.079

Table 3. Correlations among the dependent factor (averaged fitness values) and the independent factors (p_{sm}, number of clones β, similarity measure and number of neighbors k_{μ})

The averaged fitness values have a strong and positive correlation to the values of the parameter p_{sm}, for all test problems, except for F_{07} and F_{08}. In these problems, the correlation appears to be negative. Observing the third and fifth columns of the Table 3, we can see that for problems F_{07} and F_{08} the fitness values tend to be worse for smaller values of p_{sm} and also when we change from Euclidean to Hamming similarity. For those problems, this behavior is observed in Figures 13(a)-(b) and 14(a)-(b). Observing the Figure 13, we can see the dependence of the fitness values with the number of clones for F_{07}. The best values of the averaged fitness are attained using a lower number of clones. Also, we note that the fitness values are not significantly altered by the number of neighbors. This behavior becomes more evident when using Hamming similarity, as shown the contour lines of Figure 13(b). A similar behavior is observed for function F_{08}. Also, as the parameter p_{sm} decreases, the fitness values become worse, and this same behavior is verified when we change from Euclidean to Hamming similarity.

Additionally, we observe that the averaged values of the fitness are positively correlated to the number of clones for all test problems, i.e., as the number of clones decreases, the values of the fitness function become smaller.

Figures 15 and 16 display a comparison between the performance the SAAIS when using Hamming and Euclidean similarities for different values of p_{sm}. When compared to the conventional AIS (red boxplot), the results obtained by the SAAIS are better for smaller values of p_{sm} for F_{01} - F_{06} . Indeed, the results obtained using the Euclidean similarity (green boxplots) are better than the ones obtained by the Hamming similarity (yellow boxplots).

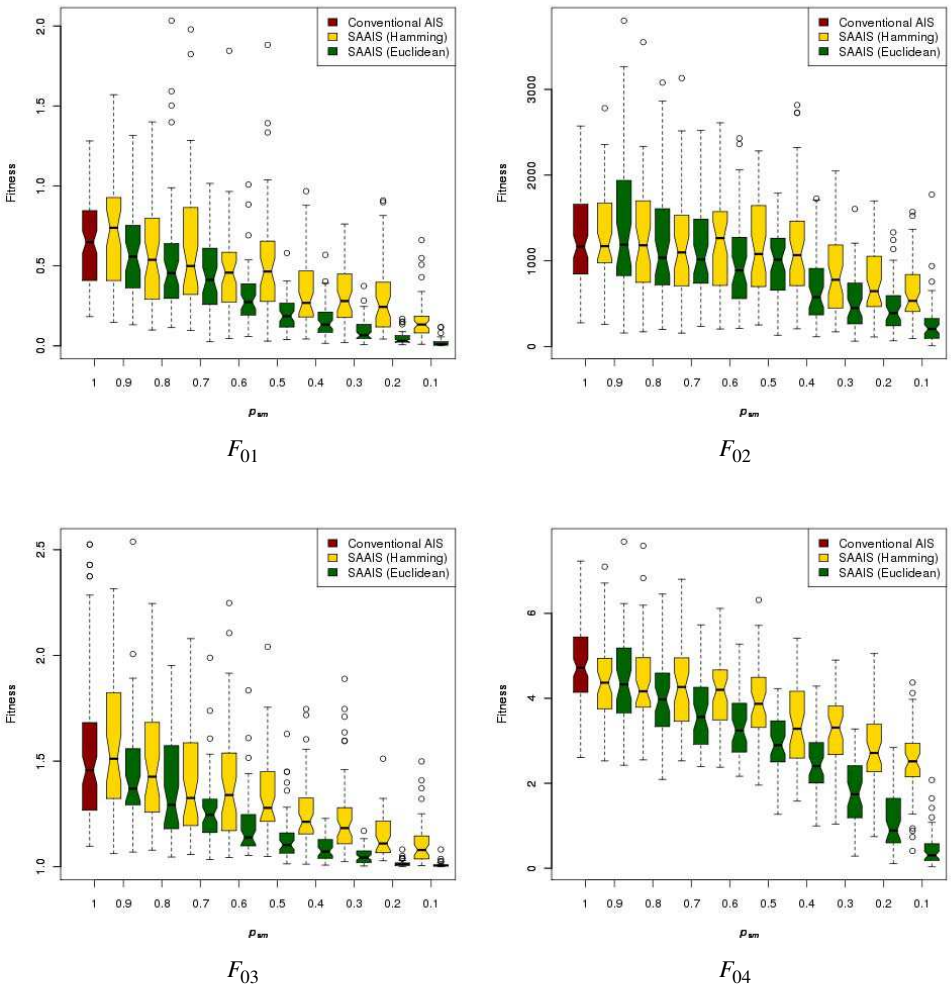


Fig. 15. Comparison of the performance of the SAAIS implementing Hamming and Euclidean similarities for different values of p_{sm}.

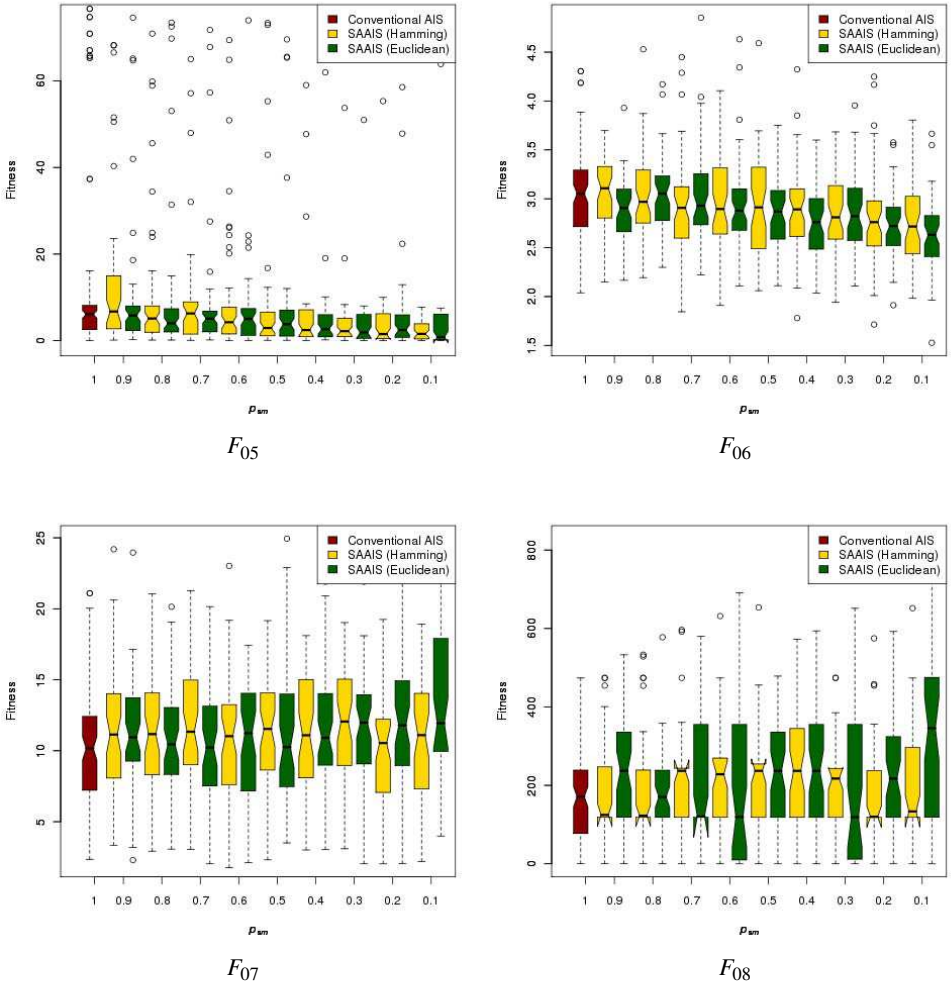


Fig. 16. Comparison of the performance of the SAAIS implementing Hamming and Euclidean similarities for different values of $p_{sm.}$

6.2 General comments about the experiments

The results found in the experiments show that the use of a surrogate model allows for increasing the total number of iterations of the algorithm and, for almost all problems considered, leads to solutions which are better than those provided by the baseline clonal selection algorithm (CLONALG).

In the Random Selection model management, the new candidate solutions are randomly chosen (with probability p_{sm}) to be evaluated exactly (via simulator). Also, the best antibodies from the population composed by the union of the current candidate solutions

and their clones are selected to form the new population. The results found in section 6.1 show that the use of this kind of surrogate model works well for most problems considered here (except functions F_{06} and F_{07}). The authors suggest that this is due to the fact that, for those functions, the surrogate model increases the exploitation of the baseline CLONALG, inducing a faster convergence to local optima. Also, we use here a very simple surrogate model, which smoothes out the fitness landscapes and has limited capabilities to approximate complex functions.

7. Concluding remarks

In this chapter we analyze the impact of introducing a Similarity-Based Surrogate Model, the k-nearest neighbors method, in a Clonal Selection Algorithm (CLONALG). The resulting framework is referred to here as Surrogate-Assisted Artificial Immune System (SAAIS).

A surrogate model is introduced in the optimization cycle by means of a simple model management (Random Selection) in order to determine which candidate solutions will be evaluated exactly via the (expensive) simulation. Thus, we increase the total number of iterations of a baseline CLONALG algorithm providing a longer evolutionary process – which may lead to improved final solutions. The total number of exact evaluations is kept constant in order to reflect the situation of a limited budget in computationally expensive real-world problems, in which each simulation can take from minutes to hours.

The results show that the new proposed SAAIS algorithm (CLONALG+SBSM) performs better than a baseline application of the clonal selection algorithm.

The underlying idea behind the use of surrogate models with AIS algorithms is to improve the exploitation capability of the latter without increasing too much its computational cost. Obviously, other ways of combining these approaches are possible, which can potentially improve even further the performance of the resulting algorithm.

8. References

Acar, E. & Rais-Rohani, M. (2008). Ensemble of metamodels with optimized weight factors. *Structural and Multidisciplinary Optimization*, Vol. 37, No. 3, pages 279- 294

Aha, D.W. (1997). Editorial. Artificial Intelligence Review, Vol. 11, No. 1-5, pages 1–6

AISWeb (2009). The online home of artificial immune systems. URL http://www.artificial-immune-systems.org, accessed June 11, 2009

Bäck, T.; Fogel, D. & Michalewicz, Z. (2000). *Evolutionary Computation 2: Advanced Algorithms and Operations*. Taylor & Francis

Bernardino, H.S. & Barbosa H.J.C. (2009). Artificial Immune Systems for Optimization, In: *Nature-Inspired Algorithms for Optimisation*, pages 389–411. Springer Berlin / Heidelberg

Blanning, R.W. (1974). The source and uses of sensivity information. *Interfaces,* Vol. 4, No. 4, pages 32–38

Bui, L.T.; Abbass, H.A. & Essam, D. (2005). Fitness inheritance for noisy evolutionary multi-objective optimization. *Proceedings of the conference on Genetic and evolutionary computation - GECCO '05*, pages 779–785, ACM, New York, NY, USA

Bull, L. (1999). On model-based evolutionary computation. *Soft Computing*, Vol. 3, No. 2, pages 76–82.

Burnet, F.M. (1959). *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press

Chen, J.H.; Goldberg, D.E.; Ho, S.Y. & Sastry, K. (2002). Fitness inheritance in multiobjective optimization. *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'02*, pages 319–326, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

Cziko, G. (1995). The Immune System: Selection by the Enemy, In: *Without Miracles*. The MIT Press

de Castro, L.N. & Zuben, F.J.V. (2000). The clonal selection algorithm with engineering applications. *Workshop Proceedings of the Genetic and Evolutionary Computation Conference – GECCO'00*, page 37

de Castro, L.N. & Zuben, F.J.V. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 3, pages 239–251

Ducheyne, E.; de Baets, B. & de Wulf, R. (2003). Is fitness inheritance useful for real-world applications? *Second International Conference on Multi-criterion Optimization*, pages 31–42, Springer

Ducheyne, E.; de Baets, B.D. & Wulf, R.D. (2007). Fitness inheritance in multiple objective evolutionary algorithms: A test bench and real-world evaluation. *Applied Soft Computing*, Vol. 8, No. 1, pages 337–349

El-Beltagy, M.; Nair, P. & Keane, A. (1999). Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations, *Proceedings of the Conference on Genetic and Evolutionary Computation – GECCO'99*, pages 196–203, Morgan Kaufmann, Orlando, USA

Emmerich, M.; Giannakoglou, K. & Naujoks, B. (2006). Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *Evolutionary Computation*, Vol. 10, No. 4, pages 421–439

Ferrari, S. & Stengel, R.F. (2005). Smooth function approximation using neural networks. *IEEE Transactions on Neural Networks*, Vol. 16, No. 1, pages 24–38

Forrester, A.I. & Keane A.J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, Vol. 45, pages 50–79

Garrett, S.M. (2004). Parameter-free, adaptive clonal selection. *Congress on Evolutionary Computation – CEC'04*, Vol. 1, 1052–1058

Giannakoglou, K.C. (2002). Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, Vol. 38, No. 1, pages 43–76

Grefenstette, J. & Fitzpatrick, J. (1985). Genetic search with approximate fitness evaluations, *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pages 112–120

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, Vol. 9, No. 1, 3–12

Jin, Y. ; Olhofer, M. & Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 5, pages 481–494

Kecman, V. (2001). Learning and soft computing: support vector machines, neural networks, and fuzzy logic models. *Complex adaptive systems*, MIT Press, Cambridge, MA, USA

Kim, H.S. & Cho, S.B. (2001). An efficient genetic algorithm with less fitness evaluation by clustering. *Proceedings of the Congress on Evolutionary Computation*, Vol. 2, No. 2, pages 887–894

Knowles, J. (2006). Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 1, pages 50–66

Lim, D.; Jin, Y.; Ong, Y.S. & Sendhoff, B. (2008). Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*

Mota, F. & Gomide, F. (2006). Fuzzy clustering in fitness estimation models for genetic algorithms and applications. *IEEE International Conference on Fuzzy Systems*, pages 1388-1395

Pilato, C.; Tumeo, A.; Palermo, G.; Ferrandi, F.; Lanzi, P.L. & Sciuto, D. (2008). Improving evolutionary exploration to area-time optimization of FPGA designs. *Journal of Systems Architecture*, Vol. 54, No. 11, pages 1046-1057

Praveen, C. & Duvigneau, R. (2009). Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design. *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, No. 9-12, pages 1087 – 1096

Queipo, N.; Arévalo C. & Pintos, S. (2005). The integration of design of experiments, Metamodeling, and optimization for thermoscience research. *Engineering with Computers*, Vol. 20, pages 309–315

Rasheed, K. ; Ni, X. & Vattam, S. (2005). Comparison of methods for developing dynamic reduced models for design optimization. *Soft Computing Journal*, Vol. 9, pages 29–37

Rasheed, K.; Vattam, S. & Ni, X. (2002). Comparison of methods for using reduced models to speed up design optimization. *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1180–1187, Morgan Kaufmann, New York

Regis, R.G. & Shoemaker, C.A. (2004). Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions Evolutionary Computation*, Vol. 8, No. 5, pages 490–505

Reyes-Sierra, M. & Coello, C.A.C. (2005). A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. *The IEEE Congress on Evolutionary Computation*, Vol 1, pages 65–72

Salami, M. & Hendtlass, T. (2003). A fast evaluation strategy for evolutionary algorithms. *Applied Soft Computing*, Vol. 2, pages 156–173

Sanchez, E.; Pintos, S. & Queipo, N. (2007). Toward an optimal ensemble of kernelbased approximations with engineering applications. *Structural and Multidisciplinary Optimization*, pages 1–15

Sastry, K.; Goldberg, D.E. & Pelikan, M. (2001). Don't evaluate, inherit. Proceedings of the Genetic and Evolutionary Computation Conference, pages 551-558, Morgan Kaufmann

Sastry, K.; Pelikan, M. & Goldberg, D.E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. *Congress on Evolutionary Computation – CEC'04*, pages 720–727

Schmidt, M. & Lipson, H. (2008). Coevolution of fitness predictors. *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 6, pages 736–749

Smith, R.E.; Dike, B.A. & Stegmann, S.A. (1995). Fitness inheritance in genetic algorithms. *Proceedings of the ACM Symposium on Applied computing – SAC'95*, pages 345–350, ACM Press, New York, NY, USA

Wanner, E.F.; Guimaraes, F.G.; Takahashi, R.H.C.; Lowther, D.A. & Ramirez, J.A. (2008). Multiobjective memetic algorithms with quadratic approximation-based local search for expensive optimization in electromagnetics. *IEEE Transactions on Magnetics*, Vol. 44, No. 6, pages 1126–1129

Zheng, X.; Julstrom, B.A. & Cheng, W. (1997) Design of vector quantization codebooks using a genetic algorithm. *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 525–30, Piacataway,NJ

Zhou, Z. ; Ong, Y.S. & Nair, P.B. (2004). Hierarchical surrogate-assisted evolutionary optimization framework, *IEEE Congress on Evolutionary Computation*, pages 1586–1593.

Evolutionary Computation

Edited by Wellington Pinheiro dos Santos

ISBN 978-953-307-008-7
Hard cover, 572 pages
Publisher InTech
Published online 01, October, 2009
Published in print edition October, 2009

This book presents several recent advances on Evolutionary Computation, specially evolution-based optimization methods and hybrid algorithms for several applications, from optimization and learning to pattern recognition and bioinformatics. This book also presents new algorithms based on several analogies and metafores, where one of them is based on philosophy, specifically on the philosophy of praxis and dialectics. In this book it is also presented interesting applications on bioinformatics, specially the use of particle swarms to discover gene expression patterns in DNA microarrays. Therefore, this book features representative work on the field of evolutionary computation and applied sciences. The intended audience is graduate, undergraduate, researchers, and anyone who wishes to become familiar with the latest research work on this field.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821