Efficient Genetic Algorithms for Solving Hard Constrained Optimization Problems
Bruno Sareni, Laurent Krähenbühl, Alain Nicolas

To cite this version:

HAL Id: hal-00135911
https://hal.archives-ouvertes.fr/hal-00135911
Submitted on 27 Mar 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Efficient Genetic Algorithms for Solving Hard Constrained Optimization Problems

B. Sareni, L. Krähenbühl, and A. Nicolas

Abstract—This paper studies many Genetic Algorithm strategies to solve hard-constrained optimization problems. It investigates the role of various genetic operators to avoid premature convergence. In particular, an analysis of niching methods is carried out on a simple function to show advantages and drawbacks of each of them. Comparisons are also performed on an original benchmark based on an electrode shape optimization technique coupled with a charge simulation method.

Index Terms—Constrained optimization methods, genetic algorithms, niching methods, shape optimization methods.

I. INTRODUCTION

One of the key features to find the optimum of hard selective functions or difficult constrained optimization problems with a Genetic Algorithm (GA) approach is the preservation of the population diversity during the search. Diversity prevents GA's to be trapped by local optima or to converge prematurely. Therefore, various procedures have been developed to avoid GA’s to rapidly concentrate their population to a single point of the search space. The first way to preserve diversity consists in protecting individuals from the loss of genetic materials by using specific mutation operators with various control schemes of mutation rates [1], [2]. On the other hand, one can also associate GA's with a niching method to avoid premature convergence [3].

This paper investigates many GA's strategies using a strong mutation operator or a niching method to solve hard-constrained optimization problems. Comparisons are carried out on an original benchmark based on an electrode shape optimization technique coupled with a charge simulation method.

II. REAL PARAMETER ENCODED GA’S WITH STRONG MUTATION OPERATORS

A Standard binary encoded Genetic Algorithm (SGA) works with a finite-length character string (chromosome) which represents the set of parameters of the problem. Typically, the chromosome of individuals is coded into a binary string. In that case, mutation is usually represented as an operation in which one of the bits on the string is flipped. For binary encoded GA's mutation rates often take small values lying between 0.001 and 0.1.

Real parameter encoded Genetic Algorithms (RGA's) are radically different from binary encoded GA's. The main structure and the selection operator are similar in both cases but crossover and mutation of RGA's directly use the parameter values of individuals to create the offspring. For example, two descendants \( x' \) and \( y' \) are obtained from two parents \( x \) and \( y \) by recombining each corresponding parameter \( i \) as follows:

\[
\begin{align*}
    x'_i &= x_i + a_i(y_i - x_i) \\
    y'_i &= y_i + a_i(x_i - y_i)
\end{align*}
\]

where \( a_i \) denotes a uniform random number in the interval [0,1]. Mutation consists generally in adding a perturbation to a design variable with a probability of \( 1/n \) where \( n \) is the number of parameters to ensure that at least one design variable is mutated for each individual. A mutated variable \( x'_i \) can be written according to (2):

\[
x'_i = x_i + \Delta x_i
\]

where \( \Delta x_i \) is the magnitude of the perturbation on the parameter \( x_i \). Typically \( \Delta x_i \) is a gaussian or a Cauchy noise such as in Evolutionary Programming or in Evolution Strategies [4], [5]. We employ another interesting mutation scheme similar to that reported in the Breeder Genetic Algorithm (BGA) [6]:

\[
\Delta x_i = (x_{i_{\text{max}}} - x_{i_{\text{min}}}) \cdot 2^{-k\alpha}
\]

where \( x_{i_{\text{min}}} \) and \( x_{i_{\text{max}}} \) denote the extreme values of the parameter \( x_i \) in the search space, \( k \) is the precision constant (typically set to 16) and \( \alpha \) is a uniform random number in the interval [0,1].

III. NICHING GA’S

A. Overview of niching GA’s

Niching methods have been developed to reduce the effect of genetic drift resulting from the selection operator in the standard GA. They maintain the population diversity and permit the GA to investigate many peaks in parallel. On the other hand, they prevent the GA from being trapped in local optima of the search space.

Niching GA’s can be classed in two different groups. The first one involves GA’s which are characterized by an explicit neighborhood since they need an explicit distance cutoff (also the similarity threshold or the niche radius) to induce emergence of niches and species in the search space. This can be an important drawback for problems for which distance between optima cannot be estimated. The second consist of techniques for which neighborhood is implicit. In that case, the algorithm requires no
information about the search space and can be easily applied to various problems without restrictions. Table 1 presents four niching methods reviewed in [3] according to the previous classification.

B. Convergence Analysis of Niching GA’s

In this section, an analysis on the behavior of various niching schemes is carried out. For that purpose, we consider the one-dimensional multimodal function defined by (4).

\[ F(x) = e^{-2\ln 2(x-0.08/0.854)^2 \sin^4(5\pi x^{3/4} - 0.05)} \]  (4)

\( F(x) \) is defined on \([0,1]\) and consists of five unequally spaced peaks of nonuniform height. Maximums are located at approximate \( x \) values of 0.080, 0.247, 0.451, 0.681, and 0.934. Maximums are of approximate height 1.000, 0.948, 0.770, 0.503 and 0.250 respectively.

To assess the efficiency of niching GA’s on this simple function, we examine the “chi-square like” performance statistic as a function of the generation number for each algorithm.

The chi-square-like performance statistics measures the deviation between the population distribution and an ideal proportionally populated distribution [9]. This criterion is computed using the actual distribution of individuals \( X_i \) and an ideal distribution mean \( \mu_i \) in all the \( i \) niches (\( q \) peak niches plus the nonpeak niche).

**chi-square like deviation** = \[ \sqrt{\sum_{i=1}^{q+1} \left( \frac{X_i - \mu_i}{\sigma_i^2} \right)^2} \]  (5)

where

\[ \mu_i = Nf_i \sum_{k=1}^{q} f_k \quad \text{and} \quad \sigma_i = \mu_i(1 - \mu_i/N) \]  (6)

for the \( i \) peak niches and,

\[ \mu_{q+1} = 0 \quad \text{and} \quad \sigma_{q+1} = \sum_{i=1}^{q} \sigma_i^2 \]  (7)

for the nonpeak niche. \( N \) denotes the population size and \( f_i \) corresponds to the fitness value of the peak \( i \). The variable \( X_i \) represents the observed number of individuals in a niche \( i \), \( \mu_i \) represents the expected ideal number and \( \sigma_i \) represents the standard deviation of the number of individuals in the ideal distribution.

The chi-square-like performance statistic characterizes the ability of the niching technique to proportionally populate the niches of the search space (the smaller the measure, the better the method). When this criterion is computed as a function of generations, it also shows how individuals evolve in the niches of the search space.

---

**Table I**

<table>
<thead>
<tr>
<th>Neighborhood</th>
<th>Niching GA’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit</td>
<td>Fitness Sharing, Clearing</td>
</tr>
<tr>
<td>implicit</td>
<td>Deterministic Crowding, Restricted Tournament Selection</td>
</tr>
</tbody>
</table>

---

**Fig. 1.** Chi-square-like deviation of the niching GA’s investigated on function \( F(x) \). The population size is \( N = 100 \).

**Table II**

<table>
<thead>
<tr>
<th>Nb of Peaks maintained</th>
<th>SH</th>
<th>CL</th>
<th>RTS</th>
<th>DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

We examine the chi-square like deviation on the function \( F(x) \) for niching GA’s reported in [3]:

- Fitness Sharing (SH) with stochastic universal selection, matching sort and a niche radius \( \sigma_s = 0.1 \).
- Clearing (CL) with stochastic universal selection, a clearing radius \( \sigma_s = 0.1 \) and a niche capacity \( k = 10 \).
- Restricted Tournament Selection (RTS) with a crowding factor \( CF = 30 \).
- Restricted Tournament Selection (RTS) with a crowding factor \( CF = 30 \).
- Deterministic Crowding (DC)

All GA’s are run with a crossover probability \( p_c = 1 \), a mutation rate \( p_m = 0.001 \) and a population size \( N = 100 \). The Euclidean distance is used in each case to evaluate the dissimilarity between individuals.

Typical chi-square deviations are displayed in Fig. 1. Table II shows the corresponding number of peaks maintained at the end of the search.

Thanks to its proportional selection operator, clearing gives a very low chi-square like deviation. It rapidly concentrates its population on the peaks of the search space and succeeds in maintaining niches. The behavior of fitness sharing is rather similar but the population is subject to noisy fluctuations that lead to an unsteady chi-square distribution.

Crowding schemes are unable to maintain low chi-square distribution during the search. The first reason for this is mentioned in [9]. Crowding methods use a replacement strategy which minimizes the changes in the population. The distribution of the population in the different niches strongly depends on the initial distribution. This explains the higher chi-square like deviations and the lower convergence noted for RTS and DC in comparison.
with those corresponding to the sharing and clearing methods. On the other hand, replacement errors can occur for individuals located at the edge of the niches [3]. For example, DC detects the five peaks of the function \( F(x) \) in the first generations. However, in the following generations, it appears that individuals located on the third peak progressively migrate to the next peak because of replacement error. At the two-hundredth generation, all individuals are discarded from the third peak yielding a poor chi-square distribution. This can be an important drawback because crowding methods might have difficulties to concentrate their population in the feasible domain for hard-constrained optimization problems.

We will verify these predictions in the next section.

IV. AN ORIGINAL BENCHMARK BASED ON THE OPTIMIZATION OF A CAPACITOR PROFILE

A. Principle

The problem consists in finding the optimal electrode shape so that the electric field is uniform on the capacitor profile from the point B to the point C [see Fig. 2(a)]. The electrode is infinite in the perpendicular direction to the Oxy plane and to the left of the point B.

This benchmark is rather interesting because an exact solution can be obtained analytically from a conformal mapping [7].

The optimal electrode profile from the point B to the point C is given as follows:

\[
\begin{align*}
  x &= 2 \ln \cos(\varphi/2) - \cos(\varphi) \\
  y &= -\varphi + \sin \varphi
\end{align*}
\quad \varphi: 0 \rightarrow \pi
\]  

The problem is solved using the electrode shape optimization method described in [8]. A geometric template has been designed to find an equivalent equipotential to the electrode shape [see Fig. 2(b)]. Eleven fictitious point charges are placed in the region limited by the internal boundary to simulate the equipotential. The optimization procedure consists in finding the optimal position and value for all charges in order to obtain an uniform electric field on the equipotential line. Consequently, the problem to be solved has 33 parameters (3 unknowns per charge). Moreover, two constraints must be fulfilled:

1. The first constraint is relative to the fulfillment of the geometric template (the equipotential must be located inside the template i.e. between the internal and external boundaries). This constraint is expressed by (9)

\[
V_{\text{max}}^\text{ext} \leq V_{\text{min}}^\text{int}
\]  

where \( V_{\text{max}}^\text{ext} \) and \( V_{\text{min}}^\text{int} \) denote the maximum potential value on the external boundary and the minimum potential value on the internal boundary respectively.

2. The second constraint requires the equipotential to be horizontal at the point B. It is represented by (10).

\[
\left| \frac{E_x(B)}{E_y(B)} \right| \leq 0.08
\]  

where \( E_x(B) \) and \( E_y(B) \) represent the electric field value at the point B and the \( x \) component of the electric field at that point respectively. This constraint ensure a horizontal profile with a maximum error of 5 degrees.

When the geometric template is violated i.e. (9) is false, the objective function to be maximized is computed by taking into account both constraints as shown in (11) at the bottom of the page where \( \lambda_1, \lambda_2 \) and \( \lambda_3 \) are penalty coefficients.

It should be noted that \( \lambda_2 >> \lambda_3 \) since the fulfillment of the first constraint has a higher order of priority in relation to the fulfillment of the second constraint.

\( \lambda_1 \) must be sufficiently large to prevent convergence on the external boundary of the feasible domain but not to high to avoid great discontinuity between feasible and unfeasible domains. In our simulations, we have set \( \lambda_1 = \lambda_2 = 10 \) and \( \lambda_3 = 10^4 \).

When the geometric constraint is fulfilled, the objective function is expressed as follows:

\[
F = \frac{1}{1 + \sigma \left( \frac{E_x(B)}{E_y(B)} \right) + \lambda_3 \max \left( 0, \left| \frac{E_x(B)}{E_y(B)} \right| - 0.08 \right)}
\]  

\[
F = \frac{1}{1 + \lambda_1 + \lambda_2 \left( V_{\text{max}}^\text{ext} - V_{\text{min}}^\text{int} \right) + \lambda_3 \max \left( 0, \left| \frac{E_x(B)}{E_y(B)} \right| - 0.08 \right)}
\]
TABLE III
COMPARISON OF VARIOUS GA SCHEMES ON THE CAPACITOR BENCHMARK
(4 INDEPENDENT RUNS—100 INDIVIDUALS—200 GENERATIONS)

<table>
<thead>
<tr>
<th>Genetic Algorithm strategy</th>
<th>$T_1$ (%) (mean)</th>
<th>$T_{12}$ (%) (mean)</th>
<th>Objective function (mean) (std-dev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGA</td>
<td>91.35</td>
<td>90.04</td>
<td>0.948 (0.028)</td>
</tr>
<tr>
<td>RGA+BGA mutation</td>
<td>83.94</td>
<td>71.48</td>
<td>0.953 (0.024)</td>
</tr>
<tr>
<td>Standard BGA</td>
<td>76.37</td>
<td>63.97</td>
<td>0.981 (0.005)</td>
</tr>
<tr>
<td>SGA+Fitness Sharing</td>
<td>36.59</td>
<td>18.33</td>
<td>0.876 (0.016)</td>
</tr>
<tr>
<td>Deterministic Crowding</td>
<td>8.36</td>
<td>0.52</td>
<td>0.888 (0.009)</td>
</tr>
<tr>
<td>Restricted Tournament Selection</td>
<td>14.45</td>
<td>3.17</td>
<td>0.918 (0.020)</td>
</tr>
<tr>
<td>SGA+Clearing</td>
<td>81.98</td>
<td>64.58</td>
<td>0.953 (0.020)</td>
</tr>
<tr>
<td>RGA+BGA mutation+Clearing</td>
<td>72.74</td>
<td>44.98</td>
<td>0.961 (0.005)</td>
</tr>
<tr>
<td>Standard BGA+Clearing</td>
<td>72.73</td>
<td>61.82</td>
<td>0.983 (0.002)</td>
</tr>
</tbody>
</table>

where $\mu_E$ denotes the mean of the electric field on the equipotential and $\sigma_E$ is the corresponding standard deviation.

The maximum value of $F$ giving the optimal profile defined by (8) is $F^* = 1$.

B. Simulations, Results and Discussion

Table II summarizes the efficiency of various GA schemes on the capacitor benchmark. All GA’s are run during 200 generations with a crossover probability $p_c = 1$, a mutation rate $p_m = 0.001$ and a population size $N = 100$. Clearing is computed with a niche capacity of 1 and a clearing radius of 0.05 (as for the niche radius in the sharing method). RTS uses a crowding factor $CF = 30$. Results in Table III are averaged on four independent runs. $T_1$ and $T_{12}$ indicate the percentage of individuals that fulfill the first constraint and both constraints simultaneously during a run of a GA.

We can notice that real-encoded GA’s (RGA and BGA) surpass the standard binary encoded GA (SGA). Niching methods are not very efficient on this unimodal problem except clearing, which improves convergence in all cases. In accordance with the predictions made in Section II, we see that crowding schemes are unable to rapidly concentrate their population in the feasible domain. In effect, RTS and DC do not create a sufficient number of individuals that fulfill both constraints simultaneously (this explains low values noted for for the $T_1$ and $T_{12}$ indicators). A standard GA coupled with fitness sharing is capable of exploring the feasible domain but fails to differentiate good solutions from bad configurations in it.

The best solution of objective $F = 0.987$ was found by the BGA coupled with the clearing method. Its characteristics are depicted in Fig. 3.

V. CONCLUSIONS

In this paper, an original benchmark consisting in finding an optimal capacitor profile with a charge simulation method has been proposed to test optimization algorithms. An analysis of niching GA’s behavior has been carried out on this problem and on a simple multimodal function to point out the convergence characteristics of each algorithm. In particular, crowding methods were unable to concentrate individuals in the feasible domain. Explicit niching GA’s such as clearing seems to be more reliable for hard-constrained problems if the niche radius can be suitably estimated.

REFERENCES