A GENETIC ALGORITHM FOR MIN-MAX PROBLEMS[1]

Helio J.C. Barbosa

LNCC/CNPQ, Rua Lauro Muller 455, 22290 160 Rio de Janeiro RJ, BRAZIL

e-mail: hcbm@lncc.br

Abstract

A family of co-evolutionary genetic algorithms(GA) is proposed for solving the min-max problem for a function $f(x, y)$. Two populations are evolved using each one an independent GA. The GA running in population $A(B)$ is a minimization(maximization) one and the individuals in this population encode values of the variable $x(y)$ belonging to the corresponding set $X(Y)$. The GA evolves for a certain number of generations on population A while the other population is kept "frozen". Then the process is applied to population B and the cycle is repeated. The fitness computation is based on the function $f(x, y)$ and the fitness of each individual in one population depends on all individuals of the other population. The results of some numerical experiments are presented.

1 INTRODUCTION

Genetic algorithms (GA's)[1, 2] are biologically inspired search procedures which have found applications in different areas of human activity. Although not originaly conceived as function optimizers[3] GAs have been shown to efficiently search complex spaces for good solutions to optimization problems. It is usually sufficient to identify the GA fitness function with the objective function to be maximized (perhaps adding a constant in order to ensure that the fitness function does not take negative values). Quite often constrained optimization problems have to be tackled and the most common procedure is to transform the constrained problem into an unconstrained one prior to applying a GA. This is usually done by adding a penalty function associated to each equality or inequality constraint. This procedure is not always efficient being quite sensitive to the choice of the (constant or variable) penalty coefficients. Numerous techniques have been proposed in the literature to deal with constraints within a GA[4] but further research is necessary in order to devise an efficient and robust procedure.

Min-max problems arise in various practical situations of interest and the objective of this paper is to present a GA for the min-max problem as well as to report the results of the initial numerical experiments performed. The paper is organized as follows: Section 2 summarizes some examples of min-max problems, Section 3 briefly reviews some of the related previous work on co-evolutionary algorithms and Section 4 describes the proposed GA. Section 5 presents some of the initial numerical results and the paper ends with conclusions an suggestions for further work.

[1]Proc. of the First International Conference on Evolutionary Computation and its Applications, pages 99-109, Moscow, Russia, 24-27 June, 1996.

2 MIN-MAX PROBLEMS

In this section some examples of min-max problems are presented. The first one, a game approach to optimization, was our initial motivation but before it could be tackled, some simpler examples of min-max problems – from the point of view of fitness computation – should be used to test the performance of the GA proposed.

2.1 A Game Approach to Structural Optimization

In structural optimization the designer is faced with the problem of setting the values of certain parameters in a way that a predefined objective function is minimized while satisfying constraints placed on the parameters themselves as well as on the performance of the structure under the set of prescribed external actions. An example would be to set the cross-sections of each member (or groups of members) of a tubular truss structure in order to minimize the total weight of the structure in a way that no member is overstressed and no node suffers excessive displacements for a set of prescribed loads. In practice, this set of prescribed external actions is developed by the designer and is supposed to somehow represent all the possible external actions applied to the structure during its entire lifetime. Kobelev[5] considers a more challenging situation where such a set is not "a priori" constructed by the designer. Instead, a game approach is taken in which a certain physically relevant quantity is introduced – the pay-off function – which the designer tries to minimize (by adjusting the available design parameters) and which nature tries to maximize (by searching for the worst possible set of external actions). The resources available to the designer are limited (the ammount of material, for example) as well as those available to nature: the set of external actions is usually a predefined bounded convex set in the load space. Nature's strategy would be to choose the load set that leads to the most severe conditions of the structure while the designer would try to find the most effective response in order to minimize the consequences of that external load set.

Let us suppose that a given linear discrete model of the static behaviour of a framed structure is written as

$$K(a)u = f$$

where K is the stifness matrix of the structure – which depends on the vector $a \in \mathbb{R}^m$ of design variables/parameters – and $u, f \in \mathbb{R}^n$ are, respectively, the vectors of nodal displacements and prescribed external loads. If the compliance of the structure is defined as the scalar value

$$c = c(a, f) = f^T u = f^T [K(a)]^{-1} f$$

one could pose the optimization problem of finding $a^* \in D \subset \mathbb{R}^m$ such that c is a minimum for any load vector $f \in A \subset \mathbb{R}^n$. From the point of view of game theory one could think of the designer choosing from his/hers set of possible design strategy set D the vector a^* in order to minimize c, having nature – the second player – as the opponent which is trying to maximize c using the "worst" (from the structure/designer point of view) vector of loads f belonging to nature's set A of possible strategies. The solution of the game would be equivalent to solving the saddle-point problem

$$\min_{a \in D} \max_{f \in A} c(a, f).$$

This application will not be considered in this paper; instead, more direct applications are introduced below and used to test the performance of the proposed GA.

2.2 Lagrange multipliers for constrained optimization

The most direct example of a min-max problem arises when one considers the introduction of Lagrange multipliers in constrained optimization such as in the standard problem

$$\min f(x) \quad \text{subject to} \quad g_i(x) \leq 0 \quad i = 1, 2, \ldots, m \tag{1}$$

for $x \in \mathbb{R}^n$. Upon introduction of the lagrange multiplier $y \in \mathbb{R}^m_+$ the lagrangian associated to this problem reads

$$L(x, y) = f(x) + \sum_{i=1}^{m} g_i(x) y_i$$

and the solution to (1) is the first element x^* of the saddle-point $\{x^*, y^*\}$ of $L(x, y)$, that is, the pair that satisfies

$$L(x^*, y) \leq L(x^*, y^*) \leq L(x, y^*) \quad \forall x \in \mathbb{R}^n, y \in \mathbb{R}^m_+$$

where $\mathbb{R}^m_+$ is the subspace of vectors in $\mathbb{R}^m$ with nonnegative components. In this way, solving the min-max problem

$$\min_x \max_{y \in \mathbb{R}^m_+} L(x, y)$$

provides the minimizer x^* as well as the multiplier y^* (which is, in some applications, as important as the primal variable x^* itself). It is assumed here that both problems are well posed in the sense that a unique pair $\{x^*, y^*\}$ exists.

Equality constraints can also be considered the only change being that the multipliers are no longer sign constrained, i.e., one has simply $y \in \mathbb{R}^m$.

2.3 Min-max approximation

Given a real function $f(x)$ on the interval $I = [a, b]$, one may be interested in finding the function $g(x)$ belonging to a predefined set $\mathcal{F}$ such that it minimizes the maximum pointwise difference between f and g over the interval I, that is

$$\min_{g \in \mathcal{F}} \max_{x \in I} |f(x) - g(x)| .$$

If each $g \in \mathcal{F}$ can be uniquely defined by a single set of m parameters $a \in S \subset \mathbb{R}^m$, then one has

$$\min_{a \in S} \max_{x \in I} |f(x) - g(a; x)| .$$

3 CO-EVOLUTION

In a standard GA the fitness function does not usually change with time so that the population of solutions can be thought of as traversing this fitness landscape and gradually converging to one of its peaks (a fitness maximum). In contrast to these static landscapes, natural evolution happens in a dynamic fitness landscape where organisms are constantly adapting to each other and to their environment[6]. Artificial coevolutionary paradigms have been used in the solution of practical problems in engineering and one idea is that of *cooperation.* Potter and De Jong[7] presented a cooperative co-evolutionary genetic algorithm for function optimization. For a problem where the values of n parameters must be found, they proposed the use of n subpopulations each of which containing competing values for a particular parameter. The fitness of a particular element of a particular species/subpopulation is an estimate of how well it cooperates with other species to produce good solutions. Two credit assignment algorithms were proposed and some results were presented. Those ideas were later applied to the co-evolution of more complex structures: neural networks[8] and sets of rules[9].

Another approach is that of *competition*, the pioneering work of Hillis[10] being particularly useful here. In[10] simulated evolution was applied to the problem of evolving minimal sorting networks for lists of a given number of elements. The fitness of each sorting network was measured by its ability to correctly solve test cases.

It was observed that the co-evolution of test cases along with the sorting networks results in an improved procedure. Two independent populations are now maintained: one of sorting networks (or hosts) and the other of test cases (parasites). Both populations evolved simultaneously interacting through their fitness function; the parasites been scored according to their ability in finding errors in the sorting networks.

More recently, Paredis[11] used a co-evolutionary approach to improve the genetic evolution of neural networks. Again two populations are maintained: A population of neural networks and a population of examples of the classification task which is submited to the neural networks. The fitness of a neural network is defined as the number of successful classifications of the twenty most recently encountered examples. On the other hand, the fitness of an example is the number of times it was incorrectly classified by the neural networks it encountered most recently. The selection of networks and examples is now biased by the fitness of each individual: fitter networks are more often selected to be tested while the more difficult examples are more often presented to the networks. However, in Paredis[11] own words: *In the strictest sense, the examples do not form a population because they are never replaced: the population consists all the time of the same 200 pre-classified examples.*

In this paper a family of co-evolutionary GAs is proposed in which two *evolving* populations, A and B, are used to solve a min-max problem of the form

$$\min_{x \in X} \max_{y \in Y} f(x, y)$$

A successful run of the GA should produce the pair $\{x^*, y^*\}$ solution of the min-max problem where x^* is the best element in population A and y^* is the best element in population B.

4 A CO-EVOLUTIONARY GA FOR THE MIN-MAX PROBLEM

Inspired by the work of Hillis[10], two populations will be evolved using each one an independent GA. The population size as well as all the GA's parameters are set independently for each population and the coupling of both evolutionary processes is made through the fitness evaluation. The basis for fitness evaluation used here is the function $f(x, y)$ where x will be taken from population A and y comes from population B. As a result, the fitness of an element in population A depends on the population B and vice-versa. The GA running in population A(resp. B) is a minimization(resp. maximization) one and the individuals in this population encode values of the variable x(resp. y) belonging to an appropriate set X(resp. Y). The fitness of an individual $x \in X$ of population A will be defined as

$$f_a(x) = \max_{y \in B} f(x, y)$$

while the fitness of an individual $y \in Y$ of population B is defined as

$$f_b(y) = \min_{x \in A} f(x, y).$$

Both GAs use a rank-based selection scheme so that no scaling/translation needs to be made and the original function $f(x, y)$ can be used for the fitness evaluation in both populations. The process consists in allowing the GA to operate on the population A for a user-defined number of generations *max_gen_a*. Then a GA is allowed to operate on population B for *max_gen_b* generations. The cycle is then repeated *max_cycles* times. This is a simplified scheme used for the numerical experiments conducted here and of course a more effective way of monitoring progress in both GAs as well as in the overall algorithm could be implemented. A pseudocode for the algorithm could be written as:

```
Algorithm
    Initialize population $A$
    Initialize population $B$
    for $k = 1, 2, ..., max\_cycles$ do
        for $i = 1, 2, ..., max\_gen\_a$ do
            generate new population $A$
            evaluate new population $A$
        for $j = 1, 2, ..., max\_gen\_b$ do
            generate new population $B$
            evaluate new population $B$
end
```

The total number of generations allowed is thus

$$ng = max\_cycles \times (max\_gen\_a + max\_gen\_b)$$

and for the case of $max\_gen\_a = max\_gen\_b$ one extreme situation would be to take

$$max\_gen\_a = max\_gen\_b = 1 \quad \text{and} \quad max\_cycles = ng/2$$

while in the other extreme one has

$$max\_gen\_a = max\_gen\_b = ng/2 \quad \text{and} \quad max\_cycles = 1.$$

The later extreme situation does not seem to be useful since population B is allowed to evolve only when population A has already possibly converged with respect to a fixed fitness landscape defined by the initial population B. The other extreme situation does not appear to be useful either as both populations explore a fitness landscape that is always changing for each generation. Intuitively, one could expect that the best performance would be achieved when each population is allowed to evolve for a certain number of generations before being "frozen" when then the other population is also allowed to evolve for a certain number of generations.

The results of some of the first numerical experiments performed so far are summarised in the next section.

5 NUMERICAL EXPERIMENTS

Unless otherwise stated, all the examples were run with a simple GA using a Gray code, a rank-based selection scheme (bias = 1.5), two-point crossover applied with probability $p_c = 0.8$, and mutation rate $p_m = 1/l$ where l is the chromossome length. An elitist procedure was adopted: the best element and a mutated copy ($p_m = 1/l$) are saved from one generation to the next. In all cases both populations have 20 individuals and, unless otherwise specified, each real parameter is encoded with 16 bits. In examples 1, 5 and 6 mid-point encoding is used while in examples 2, 3 and 4 end-point encoding was adopted. Due to the elitist procedure, the fitness of the best element in each generation of population A(resp. B) is a non-decreasing function of the generation number as long as population B(resp. A) is "frozen". When population B(resp. A) evolves the fitness of the elements in population A(resp. B) may decrease as will be seen in the numerical experiments.

The fitness of the best individual in population A in each generation (averaged in 20 runs) is shown in each example.

The legends in the figures are in the format "10 x (8+5)" meaning that those results correspond to the parameter setting: $max\_cycles = 10$, $max\_gen\_a = 8$ and $max\_gen\_b = 5$.

5.1 A first saddle-point problem

The first example is the search of the saddle-point of the simple function

$$f(x, y) = x^2 - y^2$$

over the set $S = [-1, 1] \times [-1, 1]$.

It is easy to see that the strategy for population A is to minimize the absolute value of x in order to minimize f while population B must also minimize the absolute value of y in order to maximize f. The obvious solution is the origin $(0, 0)$ where $f(x, y)$ is zero. It looks like the evolution of both populations is in fact lightly coupled in the sense that the elements in one population only *translate* the value of the fitnesses of the elements in the other population, without changing the *rank* of those elements. As a result, the evolution should not be too sensitive to the order in which the selection/reproduction process is applied to the two populations, as long as both populations are allowed to evolve for the same number of generations. Figure 1 displays the average fitness of the best element in population A in 20 runs.

5.2 A min-max problem

The second example corresponds to

$$\min_x \max_y xy$$

for points (x, y) in the set $S = [1, 4] \times [1, 4]$. One should note that the surface $z = xy$ is intrinsically the same one analysed in the previous example (only a rotation about the z axis was performed) but that there is not a geometrical "saddle" in the set S. However, from a game theoretic point of view, the problem makes sense and the strategy for the player/population A is to minimize the value of x in S in order to minimize $f(x, y) = xy$ while the player/population B should look for the highest value of y in the set S in order to maximize f.

In this case it is clear that the elements in one population only *scale* the value of the fitnesses of the elements in the other population, without changing the *rank* of those elements.

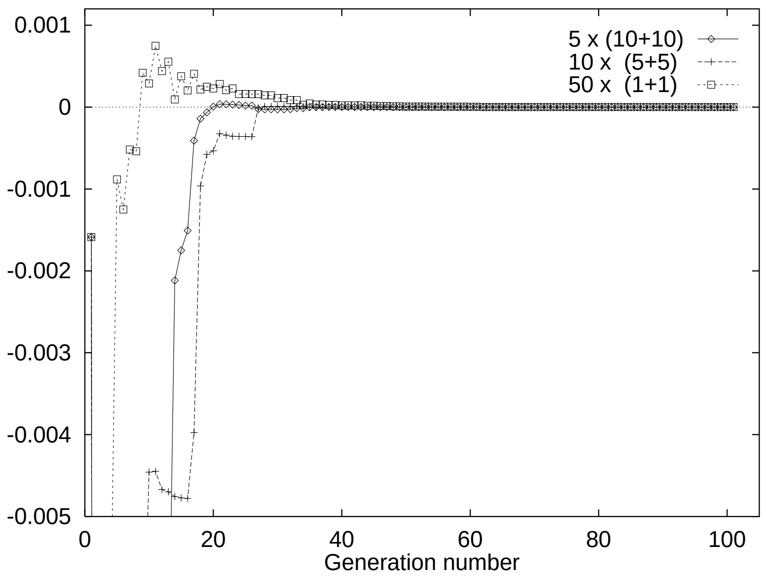

Figure 1: Average fitness of the best element in population A in 20 runs.

Because the solution lies at the vertex $(1, 4)$ of the set S, an end-point encoding is used in this example and the solution leads to $f(x, y) = xy = 4$.

The evolution is shown (Figure 2) to be similar to that of the previous example while the convergence is actually faster.

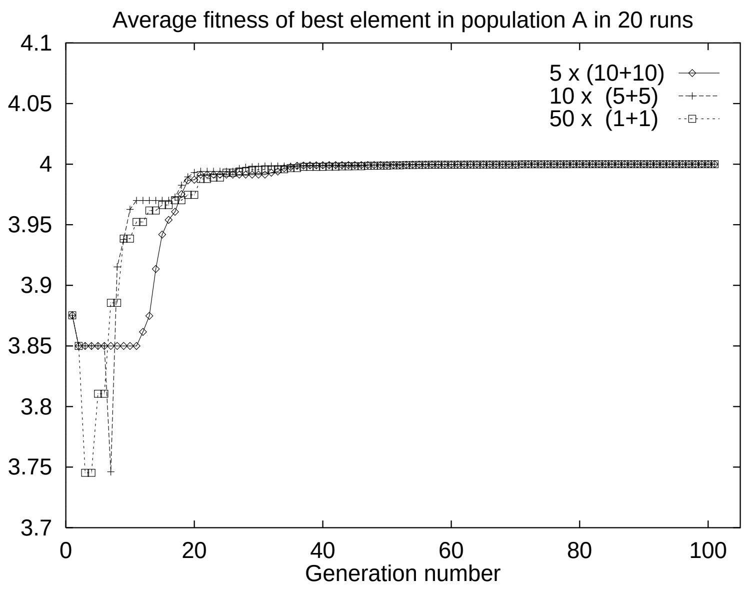

Figure 2: Average fitness of the best element in population A in 20 runs.

5.3 A third saddle-point problem

The third function to be examined here is given by

$$f(x, y) = (1.5 - x(1 - y))^2 + (2.25 - x(1 - y^2))^2 + (2.625 - x(1 - y^3))^2$$

over the set $S = [-5, 7] \times [-1, 2]$ and has a saddle-point in $(0, 1)$ where $f(x, y) \approx 14.20$. This example is found in [12]. The runs with the extreme values $max\_gen\_a = max\_gen\_b = 1$ and *max_cycles* $= 50$ were not reliable in this case (as expected) where the relation between the variables is more complex (See Figure 3.).

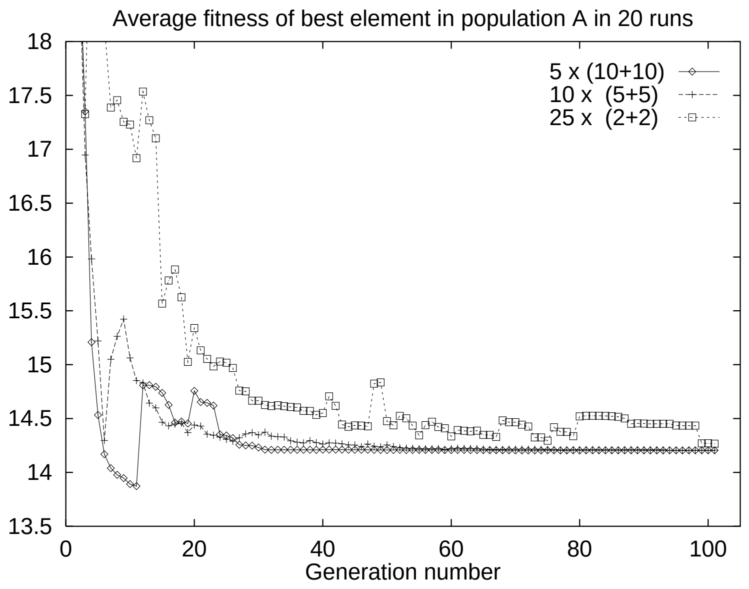

Figure 3: Average fitness of the best element in population A in 20 runs.

5.4 A first constrained minimization problem

The minimization problem[13] for the function

$$\phi(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

over the set $S = [-0.5, 0.5] \times [0, 1] \subset \mathbb{R}^2$ subject to the nonlinear constraints

$$x_1 + x_2^2 \geq 0 \quad \text{and} \quad x_1^2 + x_2 \geq 0$$

is equivalent to the search for the saddle-point of the lagrangian function

$$f(x, y) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 - y_1(x_1 + x_2^2) - y_2(x_1^2 + x_2)$$

for all $x \in S$ and all $y \in \mathbb{R}_+^2$, i.e.

$$\min_{x \in S} \max_{y \in \mathbb{R}_+^2} f(x, y).$$

The minimum is reached at the point $x = (0.5, 0.25)$ where both nonlinear constraints are inactive thus leading to null lagrange multipliers ($y_1 = y_2 = 0$) and $\phi(x) = f(x, y) = 0.25$. Figure 4 displays the performance of the algorithm for 3 parameter sets.

5.5 A second constrained minimization problem

The minimization problem[13] for the function

$$\phi(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 1)^2$$

over the set $S = [-1, 3] \times [-1, 3] \subset \mathbb{R}^2$ subject to the nonlinear constraints

$$x_1^2 - x_2 \leq 0$$
$$x_1 + x_2 \leq 2$$

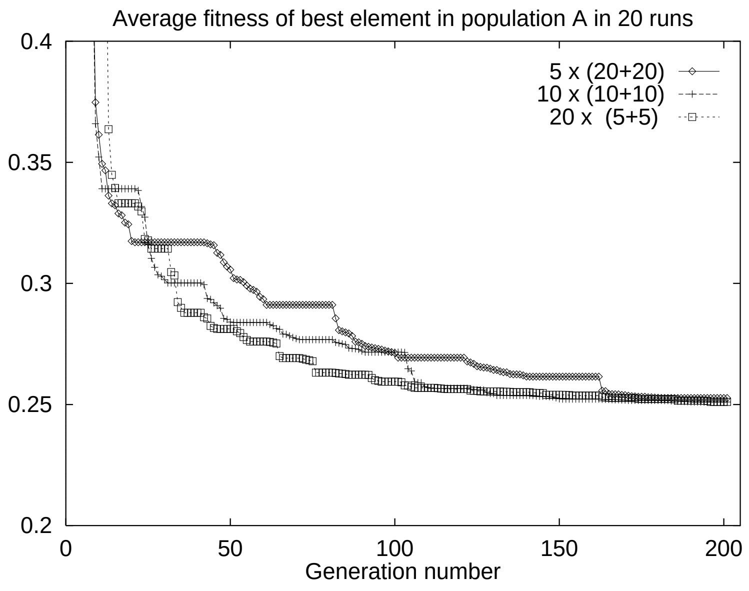

Figure 4: Average fitness of the best element in population A in 20 runs.

is equivalent to the search for the saddle-point of the lagrangian function

$$f(x,y) = (x_1-2)^2 + (x_2-1)^2 + y_1(x_1^2 - x_2) + y_2(x_1+x_2-2)$$

for all $x \in S$ and all $y \in \mathbb{R}^2_+$, i.e.

$$\min_{x\in S} \max_{y\in\mathbb{R}^2_+} f(x,y).$$

The minimum is reached at the point $x=(1,1)$ where now both nonlinear constraints are active and $\phi(x) = f(x,y) = 1$.

In Figure 5 some oscillations can be observed after 170 generations for $max\_gen\_a = max\_gen\_b = 20$.

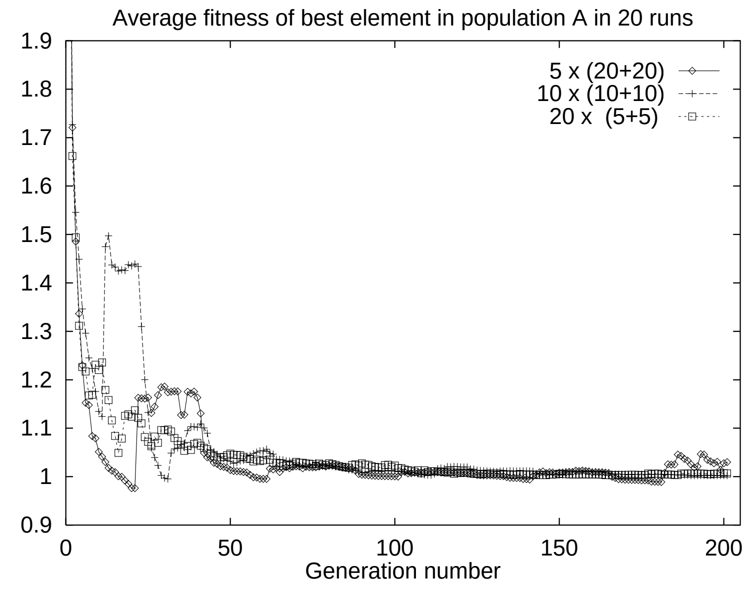

Figure 5: Average fitness of the best element in population A in 20 runs.

5.6 A min-max approximation problem

This example corresponds to finding the straight line which best approximates $f(x) = |x|$, in the min-max sense, over the interval $I = [-1, 1]$. The problem can be stated as

$$\min_{(a,b)\in\mathbb{R}^2} \max_{x\in I} \left| |x| - (ax+b) \right| .$$

In this case, each individual in population A will contain a pair of numbers, a and b, which define a straight line. Individuals in population B can have at least two possible definitions. In the first one, each individual represents a value of $x \in I$ and the population B will evolve in a way that those points which produce large deviations $\left||x| - (ax+b)\right|$ are more fit and thus, in the min-max sense, are better sampling points. However it is easy to see that for the straight line min-max approximation problem three optimal sample points exist with the same maximum value for the deviation $\left||x| - (ax+b)\right|$. In this particular example the optimal sampling points are the center and the extremes of the interval I where the deviation reaches its maximum value of 1/2. The solution is of course the line $y = 1/2$ corresponding to $b = 1/2$ and $a = 0$. The second possibility is thus to encode three sampling points per individual $t = \{t_1, t_2, t_3\}$ in population B by defining a partition of the interval I, say

$$t_1 \in I_1 = [-1, -0.3], \quad t_2 \in I_2 = (-0.3, 0.3) \quad \text{and} \quad t_3 \in I_3 = [0.3, 1]$$

and the corresponding set $T = I_1 \times I_2 \times I_3 \subset \mathbb{R}^3$. In this case the problem reads

$$\min_{(a,b)\in\mathbb{R}^2} \max_{t\in T} \left\{ \max_{1\le i\le 3} \left| |t_i| - (at_i + b) \right| \right\} .$$

The parameters $a \in [-3, 3]$ and $b \in [0, 1]$ were encoded with 16 bits each while t_1, t_2 and t_3 used 11, 10 and 11 bits respectively.

Large oscillations can be observed in Figure 6 for *max_gen_a* = *max_gen_b* = 20.

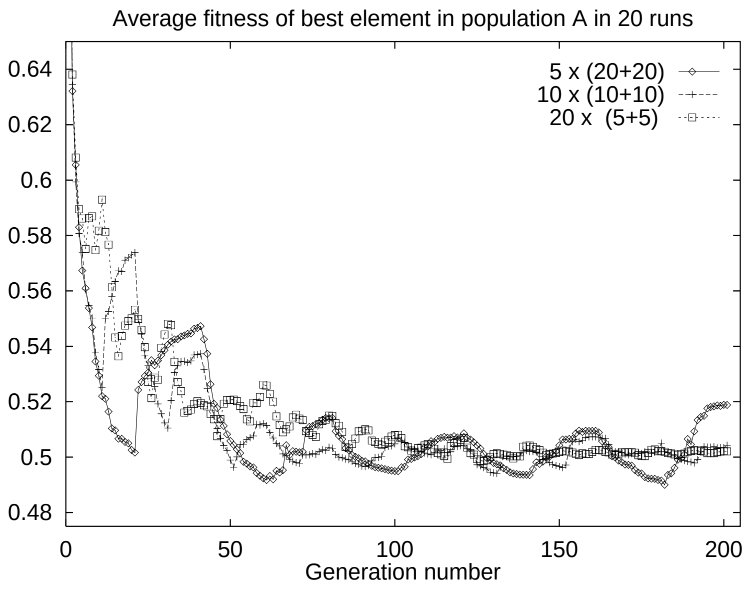

Figure 6: Average fitness of the best element in population A in 20 runs.

6 CONCLUSIONS

A co-evolutionary GA maintaining two evolving populations was proposed for the min-max problem and the results of some of the first numerical experiments performed so far were summarized. The GA running in population A(resp. B) is a minimization(resp. maximization) one and the individuals in this population encode values of the variable x(resp. y) belonging to an appropriate set X(resp. Y).

The min-max problem seems to be more delicate than some of the previous applications of co-evolutionary algorithms since the evolution/convergence of both populations is equally important in order that a good solution be found, even in the cases where one variable is, from the user's point of view, "more important".

No atempt has been made here to solve problems where more than one solution exists. In this case the techniques used in GAs for multimodal problems involving fitness sharing, clustering, etc. should be useful in the simultaneous determination of multiple solutions.

Further investigation into the design of an adaptive procedure to monitor the evolution in each population and the convergence of the whole process is necessary in order to improve the overall performance and robustness of the algorithm over the present version where the parameters *max_gen_a*, *max_gen_b*, and *max_cycles* not only remain fixed along the process but must also be "a priori" specified by the user.

Besides the types of problems tested here (saddle-points, min-max approximation and constrained optimization) the proposed GA can be applied to a game theoretic approach to optimization which remains our main motivation. Additionaly, the proposed algorithm must be tested on more challenging situations. Efforts in both directions are under way.

References

[1] J.H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, 2(2):88–105, 1973.

[2] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Reading, MA, 1989.

[3] K.A. De Jong. Genetic algorithms are NOT function optimizers. In *FOGA-92, Foundations of Genetic Algorithms*, Vail, Colorado, 24–29 July 1992.

[4] Z. Michalewicz. Genetic algorithms, numerical optimization and constraints. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms and their Applications*, Pittsburgh, PA, July 1995.

[5] V. Kobelev. On a game approach to optimal structural design. *Structural Optimization*, 6:194–199, 1993.

[6] S.G. Bullock. Co-evolutionary design: Implications for evolutionary robotics. In *Third European Conference on Artificial Life*, Granada, Spain, 4–6 June 1995.

[7] M.A. Potter and K.A. De Jong. A cooperative co-evolutionary approach to function optimization. In *Third Parallel Problem Solving from Nature*, Jerusalem, Israel, 1994.

[8] M.A. Potter and K.A. De Jong. Evolving neural networks with collaborative species. In *Proc. of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, Canada, 24–26 July 1995.

[9] K.A. De Jong and M.A. Potter. Evolving complex structures via cooperative coevolution. In *Fourth Annual Conference on Evolutionary Computation*, San Diego, CA, 1–3 March 1995.

[10] W.D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[11] J. Paredis. Steps towards co-evolutionary classification neural networks. In R. Brooks and P. Maes, editors, *Artificial Life IV*. MIT Press/Bradford Books, 1994.

[12] H-P Schwefel. *Numerical optimization of computer models.* John Wiley & Sons, Chichester, 1981.

[13] Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. In *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 98–108, River Edge,NJ, 1994. World Scientific Publishing.