Pergamon

Nonlinear Analysis, Theory, Methods & Applications, Vol. 30, No. 7, pp. 4529–4538, 1997
Proc. 2nd World Congress of Nonlinear Analysts
© 1997 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0362-546X/97 $17.00 + 0.00

PII: S0362-546X(96)00367-7

A NEW HYBRID GENETIC ALGORITHM FOR GLOBAL OPTIMIZATION

SOTIROPOULOS D.G., STAVROPOULOS E.C. and VRAHATIS M.N.

Department of Mathematics, University of Patras, GR–261.10 Patras, Greece

Key words and phrases: Global optimization, hybrid algorithm, interval arithmetic, branch–and–bound, genetic algorithms, termination criterion.

1. INTRODUCTION

We address the problem of finding reliable solutions of global optimization problems

$$f^* = \min_{x \in X^0} f(x), \tag{1.1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and the compact set $X^0 \subseteq \mathbb{R}^n$ is an n–dimensional box.

Classical gradient and random search methods behave well on simple unimodal functions but are inappropriate to be applied to difficult (and more common) problems, like non–differentiable, multimodal or noisy functions. In such cases, where traditional optimization methods fail to provide reliable results, Genetic Algorithms (GAs) can be an interesting alternative. GAs are optimization methods that evolve a population of potential solutions using mechanisms inspired from those of genetics. The choice of a "good" initial population as well as the definition of an efficient termination criterion are quite difficult tasks. Interval analysis comes to tackle these difficulties. Interval branch–and–bound algorithms are applied in order to discard from consideration large regions of the search space, where the global solution cannot exist, and to bound the global minimum.

In this paper a Hybrid Interval Genetic algorithm (HIG) is presented. The algorithm consists of two phases: In the first phase, interval arithmetic and especially an interval branch–and–bound algorithm is used to obtain small regions where candidate solutions lie. In this way, a population of potential solutions is initialized and initial bounds for the global minimum f^* are obtained.

In the sequence, a genetic algorithm is applied in such a way that all the above pieces of information are exploited. The construction of a mechanism that updates the bounds in each generation, gives the ability to define an efficient termination criterion. When the criterion is fulfilled, the algorithm converges to the global minimum f^* with certainty and extra effort can be avoided.

The contents of this paper are as follows: In Section 2 we shortly review some of the relevant material of genetic algorithms and interval analysis. Section 3 describes the proposed hybrid algorithm HIG. Numerical experiences are presented in Section 4. The final section contains concluding remarks and a short discussion for further work.

2. PRELIMINARIES

This section begins with a brief discussion about the basic concepts of GAs. For a more thorough treatment of this subject, it is recommended to see [1,2,3]. In the sequence, interval arithmetic tools which are needed for the treatment of (1.1) are established. A thorough introduction to the area of interval arithmetic can be found in [4,5,6,7].

2.1. *Genetic Algorithms (GAs)*

GAs are adaptive search and optimization methods based on the genetic processes of biological organisms. Their principles have been first laid down by Holland [8]. The aim of GAs is to optimize a problem–defined function, called the *fitness function.* To do this, GAs maintain a *population of individuals* (suitably represented candidate solutions) and evolve this population over time. At each iteration, called *generation*, the new population is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics, as, for instance, *crossover* and *mutation.* As the population evolves, the individuals in general tend toward the optimal solution.

The basic structure of a GA is the following:

ALGORITHM 1. *Simple Genetic Algorithm*

```
1.  Initialize a population of individuals;
2.  Evaluate each individual in the population;
3.  while termination criterion not reached do
    {
4.    Select individuals for the next population;
5.    Apply genetic operators (crossover, mutation) to produce new individuals;
6.    Evaluate the new individuals;
    }
7.  return the best individual
```

GAs demand only an objective function measuring the fitness of each individual. No other auxiliary knowledge such as continuity, differentiability or satisfaction of the Lipschitz condition, is required. They avoid many of the shortcomings exhibited by local search techniques on difficult search spaces since they explore new areas using knowledge accumulated during search, not randomly. They often lead to near–optimal solutions and can be easily parallelized and hybridized. In their recent work, Renders and Flasse [9] proposed hybrid methods which combine principles from genetic algorithms and "hill–climbing" methods in order to find a trade–off between accuracy, reliability and computing time.

GAs are highly depended on the choice of the *initial population* as well as on various heuristically chosen parameters. Population size ($Popsize$), mutation rate (p_m) and crossover rate (p_c) and some other parameters should be properly tuned, in order that the GA exhibits its best performance [10]. The starting population can be initialized either heuristically, by using whatever knowledge is available about the possible solutions of the specific problem, or randomly, if no such knowledge is available. Measuring the performance of a GA is not an easy task. Typically this is done by comparing the solutions found on different runs, although this implies extra amount of function evaluations. Another non trivial task is the definition of a termination criterion. A GA may terminate when a fixed number of generation has reached, when an acceptable solution is found, or when the average population fitness converges to stable fixed points. However, an efficient termination criterion is difficult to be defined. Besides, there are problems that are hard for a GA to solve [11,12]. GAs do not guarantee to find the optimal solution because (i) the search process does not ergodically cover and search the state space, and (ii) the precision limits in the encoding process can substantially reduce the solution accuracy [13].

2.2. *Interval Arithmetic*

Interval arithmetic is a generalization or an extension of real arithmetic. It has been invented by R. Moore [4] and has been used recently for solving ordinary differential equations, linear systems, verifying chaos, and global optimization.

Let $\mathbf{I} = \{[a,b] \mid a \le b,\ a,b \in \mathbb{R}\}$ be the set of compact intervals and $\mathbf{I}^n$ be the set of n-dimensional interval vectors (also called *boxes*). The interval arithmetic operations are defined by

$$A * B = \{a * b \mid a \in A, b \in B\} \quad \text{for } A, B \in \mathbf{I}, \tag{2.2}$$

where the symbol $*$ may denote $+, -, \cdot$, or $/$. The above definition is equivalent to the following rules:

$$\begin{aligned} [a,b] + [c,d] &= [a+c, b+d], \\ [a,b] - [c,d] &= [a-d, b-c], \\ [a,b] \cdot [c,d] &= [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}], \\ [a,b]/[c,d] &= [a,b] \cdot [1/d, 1/c] \ \text{ if } \ 0 \notin [c,d]. \end{aligned}$$

Throughout this paper, we denote real numbers by $x, y, \ldots$ and real bounded and closed intervals by $X = [\underline{X}, \overline{X}], Y = [\underline{Y}, \overline{Y}], \ldots$, etc. The *width* of the interval X is defined by $w(X) = \overline{X} - \underline{X}$ if$X \in \mathbf{I}$, and $w(X) = \max_{i=1}^{n} w(X_i)$, if $X \in \mathbf{I}^n$. The *midpoint* of the interval X is defined by $m(X) = (\underline{X} + \overline{X})/2$ if $X \in \mathbf{I}$, and $m(X) = (m(X_i))$, if $X \in \mathbf{I}^n$.

An *interval function* $F(X_1, \ldots, X_n)$ of intervals $X_1, \ldots, X_n$ is an interval valued function of one or more variables. $F(X_1, \ldots, X_n)$ is said to be an *interval extension* of a real function $f(x_1, \ldots, x_n)$ if $f(x_1, \ldots, x_n) \in F(X_1, \ldots, X_n)$, whenever $x_i \in X_i$ for all $i = 1, \ldots, n$. An interval function, F, is said to be *inclusion monotonic* if $X_i \subset Y_i$ $(i = 1, \ldots, n)$ implies: $F(X_1, \ldots, X_n) \subset F(Y_1, \ldots, Y_n)$.

The power of interval methods in solving optimization problems and in other applications, is exhibited in the following result due to Moore [4,5]: "*Let $F(X_1, \ldots, X_n)$ be an inclusion monotonic interval extension of a real function $f(x_1, \ldots, x_n)$. Then $F(X_1, \ldots, X_n)$ contains the range of $f(x_1, \ldots, x_x)$ for all $x_i \in X_i$ $(i = 1, \ldots, n)$*".

Inclusion functions can be constructed in any programming language in which interval arithmetic is simulated or implemented via natural interval extensions. However, computing an interval bound carries a cost of 2 to 4 times as much effort as evaluating f [7].

Interval methods for solving optimization problems consist of: (a) the main algorithm, which is a sequential deterministic algorithm where *branch-and-bound* techniques are used, and (b) accelerating devices such as cut-off test, monotonicity test, interval Newton-like step, concavity test, or local search procedures. Branch-and-bound techniques split up the whole domain into areas (branching) where bounds of the objective function f, are computed (bounding). The starting box $X^0 \in \mathbf{I}^n$ is successively subdivided into smaller subboxes in such a way that subregions which do not contain a global minimizer of f are discarded, while the other subregions are subdivided again until the desired width of the interval vectors is achieved. The development of interval tools appropriate for dealing with optimization problems is presented in [14,15,16].

3. THE NEW HYBRID INTERVAL GENETIC ALGORITHM

In this section we present our Hybrid Interval Genetic algorithm (HIG). Firstly, we give a simple model algorithm which is based on the *branch-and-bound* principle. It is used in the first part of our hybrid algorithm in order to produce boxes (with relatively small diameter ε) from which we get a "good" initial population used in the genetic portion of HIG. Secondly, we describe how the initial population is formed and propose our algorithm. Finally, we explain how a termination criterion can be defined using obtained information for bounds of f^*.

3.1. *Interval Subdivision Algorithm*

This algorithm has common features with an interval subdivision method for global optimization, but does not include local search procedures, concavity test or interval Newton-like steps, as the latter require the inclusion of the Hessian [14,16,17]. On the contrary, the cut-off and monotonicity tests are applied. *Cut-off test* uses the inclusion function F and an upper bound $\tilde{f}$ for the global minimum f^*. Boxes X with $\min F(X) > \tilde{f}$ do not reliably contain any global minimum point and, therefore, can be deleted.

Moreover, if f is differentiable then monotonicity test can be applied. *Monotonicity test* allows one to automatically recognize whether f is strictly monotone in one of the variables in some subbox $Y \subseteq X$. Let ∇F be the inclusion function of the gradient of f, ∇f. If $0 \notin \nabla F_j(Y)$ for some $j = 1, \ldots, n$, then box Y can be discarded or replaced by an edge piece [16].

The algorithm requires the following set of parameters: the initial box X^0; the inclusion function F for $f : X^0 \to \mathbb{R}$; and the maximum diameter ε of an accepted box. On output it returns a list of boxes, $\mathcal{L}$, and an interval F^* containing initial bounds for the global minimum f^*. The model algorithm is as follows:

ALGORITHM 2. *Interval Subdivision Model Algorithm*

```
1.  Set $Y = X$, $y = \min F(X)$, and $\tilde{f} = \max F(X)$ as an upper bound for $f^*$. Initialize the
    working list $\mathcal{W} = \{(Y, y)\}$ and the candidate list $\mathcal{L} = \{\}$.
2.  Choose a coordinate direction $k$ parallel to the edge of maximum length of $Y = Y_1 \times \cdots \times Y_n$.
3.  Bisect $Y$ normal to direction $k$ obtaining boxes $V^1$, $V^2$ such that $Y = V^1 \cup V^2$.
4.  Calculate $F(V^1)$ and $F(V^2)$. Set $v^i = \min F(V^i)$ for $i = 1, 2$. Improve the upper bound
    $\tilde{f} = \min\{\tilde{f}, \max F(V^1), \max F(V^2)\}$.
5.  Remove $(Y, y)$ from the working list $\mathcal{W}$.
6.  Cut-off test: discard the pair $(V^i, v^i)$ if $v^i > \tilde{f}$, for $i = 1, 2$.
7.  Monotonicity test: discard the remaining pair(s) $(V^i, v^i)$ if $0 \notin \nabla F_j(V^i)$ for any $j \in \{1,
    2, \ldots, n\}$, and $i = 1, 2$.
8.  If $w(F(V^i)) < \varepsilon$ then insert the pair $(V^i, v^i)$ to candidate list $\mathcal{L}$; else insert it to the working
    list $\mathcal{W}$. The insertion is done in such a way that the second members $v^i$ of all pairs do not
    decrease.
9.  If the list $\mathcal{W}$ becomes empty, then set as lower bound, $\underline{F^*}$, the second member of the first
    element of list $\mathcal{L}$, and as upper bound, $\overline{F^*}$, the current $\tilde{f}$. Return list $\mathcal{L}$ and interval $F^*$
    containing the bounds.
10. Denote the first element of $\mathcal{W}$ by $(Y, y)$. Go to Step 1.
```

The above sequential deterministic algorithm has been mainly established to produce a list of relatively small boxes containing various stationary (minima, maxima or saddle) points. According to our verifying procedure the global minimizer exists with certainty in one of these boxes. Of course, if the inclusion function gives the range of f in a particular box, then the box with the minimal lower bound in the list $\mathcal{L}$ contains the global minimizer.

In order to discard additional regions, any local (non-interval) optimization method that pursues the aim of delivering small function values at the first stages of Algorithm 2 can be used. In this way, cut-off test (Step 5) is more effective and regions containing various useless stationary points can be discarded. Cut-off test does not require additional pieces of information, but it may decrease the space complexity, i.e. the maximal length of working list $\mathcal{W}$. Evidently, the monotonicity test (Step 6) can be applied when the function f is differentiable. If this is not the case Step 6 must be removed. Additionally, in many cases a huge amount of time can be gained by applying different strategies in Step 1, instead of bisecting a box orthogonal to the direction with greatest diameter. Various strategies and confirmation of this effect are recently proposed by Csendes and Ratz [18,19]. Also, related approaches and implementations of Algorithm 2 can be found in [17].

Furthermore, Algorithm 2 gives an additional information regarding the bounds of f^*. These bounds are fundamental for the construction of our termination criterion (explained later) used in the genetic portion of HIG.

3.2. HIG Algorithm

In the second phase, a genetic algorithm is applied. As stated before, boxes obtained by Algorithm 2 are used to form the initial population of a GA. This initialization can be done as follows: Firstly, the population size, *Popsize*, is defined. The midpoints of the boxes in list $\mathcal{L}$ are taken as members of the initial population. In this way, the number of individuals is equal to the number of the above boxes, $\#\mathcal{L}$. If $\#\mathcal{L}$ is greater than *Popsize*, then *Popsize* is replaced by the value $\#\mathcal{L}$. If it is smaller, the population is increased by taking sequentially a box from the list $\mathcal{L}$, randomly selecting a point within it, and adding this point to the population. This procedure takes place cyclically until the number of individuals reaches the value *Popsize*.

The number of boxes contained in $\mathcal{L}$ depends on the choice of the heuristic parameter ε of Algorithm 2. For all the problems tested a value of $\varepsilon \in [0.001, 0.1]$ returns a list length value $\#\mathcal{L}$ smaller than $Popsize = 50$. In general, according to our experience, the choice of ε is proportional to the diameter of the initial region and the morphology of the objective function.

Next, we combine Algorithms 1 and 2 to obtain the following hybrid algorithm:

ALGORITHM 3. *Hybrid Interval Genetic Algorithm, HIG*

```
1. Apply an interval subdivision algorithm;
2. Initialize the population;
3. Evaluate each individual in the population;
4. while termination criterion not reached do
   {
5.   Update the bounds;
6.   Select individuals for the next population;
7.   Apply genetic operators to produce new individuals;
8.   Evaluate the new individuals;
   }
9. return $f^*$ and $x^*$.
```

The bounds $\underline{F^*}$ and $\overline{F^*}$ which are obtained by the Step 1 of the above algorithm satisfy the following relation:

$$\underline{F^*} \leq f^* \leq \overline{F^*}.$$

Since, in general, we choose a relatively large value of ε and also the inclusion function is actually the natural interval extension of f, the $\underline{F^*}$ and $\overline{F^*}$ are overestimated. The basic idea for the termination criterion referred to Step 4 is to make these bounds of f^* sharper at each generation.

Now, although it is always possible for a genetic algorithm to update and improve the upper bound at each generation, it is impossible to give a better lower bound. Also, the lower bound is much smaller than the global minimum value. GAs are not able to provide a safe mechanism for updating the lower bound since they sample the objective function at only a finite number of points.

Thus, we update the bounds in the following way: At each generation, the upper bound $\overline{F^*}$ of the global minimum is replaced by the minimum of the current upper bound and the best individual's performance. That is,

$$\overline{F^*} = \min\left\{\overline{F^*},\ \text{BestPerformance}\right\}.$$

Updating the lower bound is not an easy task. To do this, we utilize interval arithmetic and the notion of the *current shrinking box* which is defined in the sequel.

A current shrinking box, denoted by X_S, is the smallest convex interval vector containing a subset S of n–dimensional individuals $x_1, x_2, \dots, x_k$, where $k \le$ *Popsize*, whose performance is within the interval $[\underline{F^*}, \overline{F^*}]$.

The shrinking box is constructed when the number k of individuals with performance within the current interval $[\underline{F^*}, \overline{F^*}]$ exceeds a predefined number, r, which is proportional to the total population size, *Popsize*. According to nature's survival–of–the–fittest principle, the number k will certainly exceed, r, at some generation. Of course, the indication of the construction of the shrinking box can be handled as a convergence test of a GA. If it is not constructed, the algorithm does not converge.

When a shrinking box X_S is constructed, an estimation of the range of f over X_S is obtained using interval arithmetic. In this way, new bounds $\underline{F_S}$ and $\overline{F_S}$ are obtained. Thus, the current bounds of the global minimum are updated as follows:

$$\overline{F^*} = \min\{\overline{F^*}, \overline{F_S}\}, \qquad \underline{F^*} = \max\{\underline{F^*}, \underline{F_S}\}.$$

Evidently, as $w(X_S)$ tends to zero the individuals are accumulated to a point. Additionally, if $w(F^*)$, $F^* = [\underline{F^*}, \overline{F^*}]$ tends to zero, which means that $\underline{F^*} \simeq \overline{F^*} \simeq f^*$, then this accumulation point is a global minimizer of f.

Based on this, our algorithm proceeds until the following relations hold:

$$w(X_S) < \varepsilon_x \quad \text{and} \quad w(F^*) < \varepsilon_F,$$

where ε_x and ε_F are the tolerances for x^* and f^* respectively.

According to our opinion the above termination criterion is very effective compared with other widely used criteria and by using this, extra computational effort is saved.

4. NUMERICAL EXPERIENCES

The numerical tests have been carried out on an 80486/133MHz PC IBM compatible using an implementation of the HIG algorithm in C-XSC which is a C++ class library for scientific computing with automatic result verification [17]. The inclusion functions have been produced by natural interval extensions.

The performance of HIG algorithm is measured according to the Expected Number of Evaluations per Success performance index (ENES). This number has been defined at the IEEE International Conference on Evolutionary Computation (ICEC'96), May 20–22, 1996, Nagoya, Japan. Details on this can be found from the home page `http://iridia.ulb.ac.be/langerman/ICEO.html`. *ENES* represents the mean number of function evaluation needed in order that the HIG algorithm reaches the termination criterion and it is computed by running twenty independent runs of the algorithm with the same parameters, until the termination criterion is fulfilled. If *NS* is the number of successes, that is the number of runs that termination criterion is reached and *NE* is the total number of function evaluations during the 20 runs, the ENES is defined as *ENES= NE/NS*. If the desired value for the global minimum can never be reached, then *ENES* is not defined.

We have selected difficult optimization problems for both Interval Analysis and Genetic Algorithms, and experimental results are exhibited in the sequel. HIG algorithm has been compared with GENESIS 5.0 optimization system due to Grefenstette [20]. Two different sets of runs of GENESIS have been made: GENESIS-RP and GENESIS-HP, where the initial population has been initialized randomly and heuristically (the same as HIG initial population), respectively. For each test problem we have executed twenty independent runs. Both HIG and GENESIS have been provided with the same set of parameters. GENESIS terminates when a predefined number of trials (function evaluations) is reached. Assuming that one interval evaluation is equivalent to two floating-point evaluations, the number of total trials supplied to GENESIS has been computed by the formula: $TT = 2*(IFE + IGE) + MNE$, where *IFE* is the total number of interval function calls to determine the range of the function, *IGE* is the total number of interval gradient evaluations, and *MNE* is the maximum number of real function evaluations of a particular run, after twenty runs of HIG.

For the following test problems, further reported parameters are: n the dimension of the problem, X^0 the starting box, x^* the global minimizer, and f^* the global minimum. BV is the best value each algorithm has reached, and $TOL=\varepsilon_x=\varepsilon_F$ the error tolerance for approximating x^* and f^*.

PROBLEM 4.1 *Levy function* ($n = 2$) [18]. This function is defined by

$$f(x) = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \sum_{j=1}^{5} j\cos[(j+1)x_2 + j] + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2,$$

within the initial box X^0 specified by $-10 \le x_i \le 10, i = 1, 2$. The global minimum is $f^* = -176.1375$ at $x^* = (-1.3068, -1.4248)$. There are about 760 local minima in the minimization region. The large number of local optimizers makes it extremely difficult for any approximation method to find the global minimizer.

All genetic algorithms have run for the same set of parameters. That is: population size $Popsise = 50$, crossover rate $p_c = 0.6$, and mutation rate $p_m = 0.01$. Algorithm 2 has returned 25 boxes and has found that f^* belongs to the interval $[-191.8058, -175.0057]$, with total effort $IFE = 292$ and $IGE = 178$. We have executed the HIG algorithm for twenty independent runs and we have found that the maximum number of real function evaluations of a particular run has been $MNE = 2650$. According to the previous formula, the total number of trials for both GENESIS-HP and GENESIS-RP is $MNE \equiv TT \approx 3600$.

The results exhibited in Table 1 clearly show that HIG has been the only algorithm that has found the global minimum with certainty. HIG has succeeded in all runs and has found the optimal solution

	HIG	GENESIS-HP	GENESIS-RP
BV	−176.1375	−174.9709	−117.363658
NS	20/20	0/20	0/20
ENES	2202	—	—
MNE	2650	3600	3600
Success	100%	0%	0%

Table 1: Classification of Levy function ($n = 2$).

with the desired accuracy $TOL = 0.001$. The termination criterion has been verified in all runs. For each run the maximum number of real function evaluations has been less than or equal to 2650. The *ENES* index for HIG has been 2202, while the corresponding index for the rest algorithms cannot be defined. It is clearly seen that GENESIS-HP is superior to GENESIS-RP. This confirms our argument that the choice of a "good" initial population is crucial for the efficiency of a pure genetic algorithm.

PROBLEM 4.2 *Goldstein-Price function* ($n = 7$). [18]. This function is defined by

$$f(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times \\ \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right],$$

within the initial box X^0 specified by $-2 \le x_i \le 2, i = 1,2$. The global minimum is $f^* = 3.0$ at $x^* = (0.0, -1.0)$.

For this problem, the common set of parameters has been: *Popsise* = 50, $p_c = 0.6$, and $p_m = 0.01$. Algorithm 2 has returned 15 boxes. The initial bounds for f^* has been $[-3.4875 \times 10^6, 32.6875]$ and the total effort has been *IFE* = 60 and *IGE* = 30. As shown in Table 2, although all algorithms have found the global minimum, with *TOL*= 0.001, ENES indexes indicate that HIG outperform the rest of them. Both HIG and GENESIS-HP have succeeded in all runs, in contrast with GENESIS-RP which has succeeded only in half of them. In addition, HIG seems to be the least cost-effective algorithm, as the termination criterion has been fullfiled in all runs. For each run the maximum number of real function evaluations has been less than or equal to 1950. GENESIS-HP has exhibited better performance than GENESIS-RP, due to the good choice of initial population. However, their comparatively good performance is justified by the relatively small search region as well as by the flatness of the objective function in the global minimum's neighborhood.

	HIG	GENESIS-HP	GENESIS-RP
BV	3.000	3.000	3.000
NS	20/20	20/20	10/20
ENES	1340	1885	4089
MNE	1950	2150	2150
Success	100%	100%	50%

Table 2: Classification of Goldstein-Price function ($n = 2$).

PROBLEM 4.3 *Griewank function* ($n = 7$) [18]. This function is defined by

$$f(x) = \sum_{i=1}^{7} \frac{x_i^2}{4000} - \prod_{i=1}^{7} \cos \frac{x_i}{\sqrt{i}} + 1,$$

within the initial box X^0 specified by $-600 \le x_i \le 500, i = 1,2,\ldots,7$. The global minimum is $f^* = 0.0$ at $x^* = (0,0,0,0,0,0,0)$. It is an extremely difficult test problem since there are several thousands of local minima in this relatively large minimization region.

The common set of parameters for the genetic algorithms has been: *Popsise* = 50, $p_c = 0.6$,and $p_m = 0.001$. Algorithm 2 has returned only one box, with total effort *IFE* = 580 and *IGE* = 386.

	HIG	GENESIS–HP	GENESIS–RP
BV	0.0001	0.0005	1.0292
NS	20/20	20/20	0/20
ENES	430	2450	—
MNE	500	2450	2450
Success	100%	100%	0%

Table 3: Classification of Griewank function ($n = 7$).

The large amount of this effort, comparatively with the previous test problems, is due to the high dimension of the problem, as well as to the wide range of the search space. In Table 3, it is easily seen that both HIG and GENESIS–HP have succeeded in all the runs but, clearly, HIG has been more efficient than the latter. Observing the *ENES* indexes it is evident that an efficient termination criterion for a genetic algorithm is of great significance. HIG needs only few trials to reach the optimal solution (with $TOL = 0.001$), while the rest of them consume all the trials and terminate without any guarantee that the global minimum has been found. GENESIS–HP has performed well (by taking in advantage its good initial population), in contrast with GENESIS–RP which has been completely misleaded by the large search region and the enormous number of local minima. Of course better results for GENESIS–RP can be obtained by tuning the size of the randomly selected population.

5. CONCLUSIONS AND FURTHER WORK

In this contribution, we present a hybrid genetic algorithm for finding guaranteed and reliable solutions of global optimization problems. This algorithm uses the branch–and–bound principle to obtain small regions where candidate solutions lie. In this way, a highly–performing initial population is formed and initial bounds for the global minimum f^* are obtained. By applying a genetic algorithm using this population as well as a safe and reliable technique for updating properly the bounds of f^*, we are able to compute with certainty global minima for various difficult test problem.

The proposed algorithm becomes more effective when a new termination criterion is used. This criterion is based on the notion of a shrinking box and using this extra computational effort is avoided.

HIG has exhibited high performance when applied to difficult problems, especially to multimodal and high–dimensional objective functions. It has been clear that HIG outperforms traditional genetic algorithms. Also, for all the problems examined, HIG has given better results for both Algorithm 1 and 2 studied separately.

In its present form, HIG gives us one global minimum. Assuming that clusters of global minima do not exist, HIG can give all the global minimizers, using dynamically produced subpopulations. As a future work, we are going to investigate the construction of a pure genetic algorithm whose genetic operators will be based on interval arithmetic. In this way and using only function evaluations, we hope that all the global minimizers will be computed with certainty.

REFERENCES

1. GOLDBERG D.E., *Genetic Algorithms in search, optimization and machine learning*, Addison-Wesley, Mass., (1989).
2. DAVIS L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, (1991).
3. MICHALEWICZ Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, (1996).
4. MOORE R.E., *Interval Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, (1966).
5. MOORE R.E., *Methods and Applications of Interval Analysis*, SIAM Publ., Philadelphia, (1979).

6. ALEFELD G. & HERZBERGER J.,*Introduction to Interval Computations*, Academic Press, New York, (1983).
7. NEUMAIER A., *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, (1990).
8. HOLLAND J.H., *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Mass., (1975).
9. RENDERS J. & FLASSE S., Hybrid Methods Using Genetic Algorithms for Global Optimization,*IEEE Trans. Syst., Man, Cybern.*, **26**, 243-258, (1996).
10. GREFENSTETTE J.J., Optimization of Control Parameters for Genetic Algorithms,*IEEE Trans. Systems, Man, and Cybernetics*, **16**, 122-128, (1986).
11. FORREST S. & MITCHELL M., What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation, *Machine Learning*, **13**, 285-319, (1993).
12. MAGOULAS G.D., VRAHATIS M.N. & ANDROULAKIS G.S., Effective backpropagation training with variable stepsize, *Neural Networks*, **9**, No. 6, (1996), in press.
13. INGBERG L. & ROSEN B., Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison, *J. Mathematical and Computer Modeling*, **16**, 87-100, (1992).
14. RATSCHEK H. & ROKNE J., *New Computer Methods for Global Optimization*, Ellis Horwood Limited, (1988).
15. RATSCHEK H. & ROKNE J., Interval Tools for Global Optimization, *Computers Math. Applic.*, **21**, 41-50, (1991).
16. HANSEN E., *Global Optimization Using Interval Analysis*, Marcel Dekker Inc., (1992).
17. HAMMER R., HOCKS M., KULISCH U. & RATZ D., *C++ Toolbox for Verified Computing*, Springer-Verlag, (1995), see also `http://www-iam.mathematik.uni-karlsruhe.de/html/language`.
18. RATZ D. & CSENDES T., On the Selection of Subdivision Directions in Interval Branch-and-Bound Methods for Global Optimization, *J. Global Optimization* , **7**, 183-207, (1995).
19. CSENDES T. & RATZ D., Subdivision directions selection in interval methods for global optimization. To appear in SIAM Journal of Numerical Analysis.
20. GREFENSTETTE J.J., A User's Guide to GENESIS, Version 5.0, (1990).