ON AN EXPERIMENT IN SOLVING EQUATIONS USING GENETIC ALGORITHMS

Márton-Ernö BALÁZS*

Received: January 15, 1993
AMS Subject Classification: 68T01

REZUMAT.- Asupra unui experiment de rezolvare a ecuaţiilor folosind algoritmi genetici. Articolul prezintă concluziile unui experiment de utilizare a unui algoritm genetic pentru rezolvarea ecuaţiilor. Rezultatele sînt comentate în paralel cu prezentarea paşilor unui algoritm genetic general. Sînt de asemenea arătate unele soluţii specifice clasei de probleme abordate precum şi sugerate posibilităţi de îmbunătăţire ale rezultatelor.

0. Introduction. Solving equations has been for a long time a central problem in numerical applications. The most important keywords in the field of equation solving are the *domain of convergence, rate of convergence* and *complexity* of the methods. Although there are many nice results in developing methods of rather high rate of convergence with acceptable complexity, most of them are strongly limited in what concerns their domain of convergence. Nevertheless in many such methods testing for the convergence domain is of a comparable complexity to the computation itself. Another inconvenience of most of these methods is that of converging to a single solution (that is obviously induced by their nature) whereas there might be many other viable solutions around. These deficiencies of the classical equation solving methods gave place to the searching for methods with "large" convergence domains (see [1]). Even if these type of methods are usually of a great complexity they may be used at least for isolating to a certain amount the solutions,

* *"Babeş-Bolyai" University, Department of Mathematics and Computer Science, 3400 Cluj-Napoca, Romania*

giving then place to the "specialized" fast converging ones. Beside this approach in many cases these mathods can be improved by using some kind of domain specific properties as shown in [5]. We also should mention that the developing of these "unorthodox" methods was greatly encouraged by the development of computing techniques towards parallel processing. In the present paper we shall make some remarks on an experience in solving equations using *genetic algorithms*.

1.A short introduction to genetic algorithms. One of the striking ideas for designing optimum searching methods is that of using principles of organic evolution [3].The approaches emerging from this idea (*genetic algorithms* and *evolution strategies*) rely upon imitating the adaptation and selection capacity of natural populations. Although the two main approaches are similar to a large extent in the followings we shall discuss in the terms of genetic algorithms.

1.1. General structure of genetic algorithms. Omitting the details we can say that genetic algorithms (GA) operate on a fixed size *population* of individuals of the same type creating new *generations* of it in successive cycles.

Analogously with natural populations the creation of a newgeneration is directed by a *fitness criterion* which conditions the perpetuation of the most viable individuals. In GA-s this is carried out through the following operations:

a.) *evaluation of the fitness* of each individual of the

current population;

b.) *formation of a gene pool* built up from the most fit individuals;

c.) creation of the new generation by *recombination* and *mutation* of elements in the gene pool.

The process of creating new generations is iterated until the average fitness of the population ceases to improve, by which time the population has converged to a few kinds of well performing individuals.

As stated above the creation of a new generation is carried out by recombination and mutation of elements in the current gene pool. Individuals contribute to the gene pool in a number depending on their fitness. The elements of the gene pool are codings of the best performing individuals (according to the fitness criterion) of the current population. Such a coding has to be a condensed representation of individuals which captures their defining features.

Recombination is performed through the crossover operator which randomly selects a fixed *number of parent genes* from the gene pool and creates an *offspring gene* by combining a randomly chosen part of them. Usually this combination is done either by combining the "tails" of the parent genes (one point crossover), or by combining some internel part of them (two point crossover). We should mention here that this crossover is not merely a random creation of a new gene since it preserves features of the parent genes.

Mutation is meant for preserving the diversity of the

population which prevents the algorithm from converging to local solutions as well as from omitting some solutions in the search space. It is esentially carried out by randomly changing some portion of a gene.

Now we can present an outline of a GA (after [3] and [5]). Let N be the fixed size of the population and f the function which measures the fitness of an individual (*fitness function*).

```
procedure GA
  begin
    t:=0;   { t is the number (moment) of the generation }
    create an initial population P*t(;
    evaluate the fitness of each individual in P*t( using f;
    while termination condition not satisfied do
      begin
        create a gene pool G*t( bz coding the most fit
                individuals of P*t(;
        t:=t+1;
        select parent genes from P*t-1(;
        recombine selected genes and build P*t(;
        evaluate the fitness of each individual in P*t(
                using f;
      end
  end.
```

In the followings we shall make our references to this description.

1.2. A short overview of some results concerning GA-s. We

shall mention the results and make some comments following the steps in the algorithm outlined in the previous section. *Creation of the initial population*. Although it is not explicitly required most impelmentations of genetic algorithms begin with a uniformly distibuted random initial population. This approach is reasonalble since it intuitively prevents a possibly long-drawn phase of diversification of individuals before convergence, but in some specific problems it has been shown that it is not absolutely necessary. *Evaluation of fitness of individuals*. The most common way of treating fitness evaluation is by simple function evaluation. To do this the objective function o (the function to be optimized) has to be transformed by composition to a nonnegative number: $f(.)=u(o(.))$. This transformation may damage some desirable properties of the GA (see [5]).

Creation of the gene pool. In the creation of the gene pool one should handle the following problems: the selection of individuals from the current population, establishing the *contribution* of each selected individual to the gene pool and *coding* of the resulting set of individuals.

Let us start with the *coding*. As it results from the literature many contributors to GA-s' theory argue for binary coding but there are some who consider other representation more suited for special classes of problems [5]. Choosing binary coding, beside other features is apealing for its simplicity in implementing recombination on digital computers.

As already stated *selection* and establishing the

contribution of individuals to be used for building the gene pool depends on the fitness computed in the previous step of the algorithm. There are several selection algorithms (which also establish the contributions) of which the most used seems to be *the proportional selection algorithm* which computes the number of contributions of an individual as ratio of the individual's fitness and the average fitness of the current population. For this selection algorithm Holland gave a characterisation of the number of classes of individuals in the next generation (the Schema Theorem, see [5]). We omit further discussions here.

Selection of parent genes. Parent *genes* are selected randomly from the gene pool. The number of parents used for generating an offspring gene depends on the model used. As presented in [5] GA-s may use either a *one parent* or a *manyparents* model.

Recombination of genes. We have already mentioned that recombination of genes is performed through *crossover* and *mutation*. Crossover means combination (tipically exchanging) of a randomly chosen part of the parent genes. This operation is the main mechanism which creates new candidates for solution deriving them from old ones by preserving a significan part of their properties. In order this latter statement to be true the size of gene portions to be combined is kept small. In the previous section we gave the two alternative crossover types: *one point* and *two point* crossover.

It is not our goal to go in more details about GA-s here, the interested reader is suggested to read [5].

2. Notes on an experiment in solving equations. The main goal of the present paper is to report on some notes of the author made during an experiment in using a GA for solving various equations. In the followings we shall present some of them.

Our purpose was to try to develop a GA type method for approximatively solving equations of the form $f(x)=0$, where $f:R^n \to R$. Obviously this problem can easily be transformed into another, suited for application of a GA: *Approximate the minimum points of function* $g:R^n \to R$ where $g(x)=|f(x)|$ of which only those solutions x^* are solutions of the original problem for which $g(x^*)=0$. So we applied a GA to this problem. For the beginning we only studied one dimensional problems which also raised some difficulties.

In our implementation the *population was made up* of real numbers of the domain in which solutions were sought (i.e. an interval of the real axis). We varied the *population size* in different runnigns ranging from 10 to 200 individuals, noting that the convergence of the algorithm improved for population sizes over 100, by which time it correctly made distinction between more solutions.

Whether the the *initial population* was built by uniformly distributing numbers on the search segment or not seemed in our experiment of no significance - the number of steps in which the desired precision was achieved was about the same.

The *coding* used is a two level one which first mappes the search domain into the segment [0,1], then considered the first

k significant binary digits of the representation, where k depends on the desired precision. This coding is a simple one to one mapping from individuals to their representation which also has the advantage of easy reconstruction of an individual from a code. As all binary codings it also has the advantage of simplicity of implementing crossover.

For *selection* the proportional selection algorithm was used, with randomly distributing the unassigned positions in the gene pool to individuals in the current population. The only model tried was the two parents one.

The most problems in tuning the algorithm were raised by the recombination of genes and building of the new population.

Implementing *crossover* and *mutation* was thechnically speaking simple due to the binary coding used. A two point crossover was tried with length ranging from 1 to 8 bits of the representation. Here we can make the following notes:

a.) The *convergence speed of the algorithm* was improved by making the probability with which the portion of genes to be combined was selected depend on the generation number in such a way that in "advanced" generations less significant digits were combined with greater probability than the more significant ones. This observation sounds intuitively fair since in advanced stages of the search the algorithm should work harder on improving the precision of the already approximated solutions but without excluding the possibility of finding new candidates.

b.) When selecting a parent gene with a probability p it is essential *to select the coparent gene* with a probability $1-p$

in order the two parents to be different. Probably this note isnot valid if crossover is done with other combination than simple exchanging.

For *building the new population* from the gene pool many variants have been tried.

At the beginning the new population was entirely constructed from the gene pool. With this approach for all the equations tried the method converged to just one (of the known) solutions even if in early stages of search all the solutions have had some approximation.

This fenomenon was eliminated when e generation was built up by preserving the most fit individuals from the previous one and generating just some new individuals from gene pool. This is however a known practice [5] even if we do not have sufficient data to support the *one fifth principle*.

In the figure below we present the outlines of the results obtained with our implementation for one of the equations studied.

Evolution of number of individuals (real numbers) around the solutions of equation **sin 2x** over **(1.4)** for a population size of 100.

	1.57...	3.14...	Total
Generation: 1	7.00%	7.00%	14.00%
Generation: 2	17.00%	26.00%	43.00%
Generation: 3	41.00%	8.00%	49.00%
Generation: 4	32.00%	14.00%	46.00%
Generation: 5	33.00%	32.00%	65.00%
Generation: 6	47.00%	20.00%	67.00%
Generation: 7	59.00%	25.00%	84.00%
Generation: 8	64.00%	29.00%	93.00%
Generation: 9	69.00%	24.00%	93.00%
Generation:10	62.00%	31.00%	93.00%
Generation:11	65.00%	32.00%	97.00%

3. Conclusions. From the experience presented in the previous section we conclude that GA-s can be successfully used for solving equations although much care has to be taken in designing the implementation. The expected advantage of this approach is that of being applyabble even for equations which are defined by non continuous functions (not to speak about not differentiable ones), a quality which is not common for other known methods.

Obviously the implicit parallelism of the method is an other appealing property which should be exploited in implementations. As we did it in [1] we suggest the combining of the GA with specialized, fast solving methods in order to obtain both generality and speed.

In the followings we intend to work on improving the selection methods and building of the fitness function using, where possible properties of the equation defining function.

REFERENCES

1. M.E.Balázs - A *Heuristic Search Type Algorithm for Solving Nonlinear Equation Systems*, Studia Univ. "Babeş-Bolyai" Mathematica, XXXIII, 3, 1988.
2. J.J.Grefenstette, J.E.Baker - *How Genetic Algorithms work: A Critical Look at Implicit Parallelism*, Proceedings of ICGA'89, ed. J.D.Schaffer, (1989).
3. F.Hoffmeister - A *Survey of Evolution Strategies*, roceedings of ICGA'91, ed. R.K.Blew, (1991).
4. J.H.Holland - *Adaptation in Natural and Artificial Systems*, University Michigan Press (1975).
5. G.E.Liepins, M.R.Hilliard - *Genetic Algorithms: Foundations and Applications*, Annals of Operation Research, 21 (1989).