Solution of Simultaneous Non-Linear Equations using Genetic Algorithms

Angel Fernando Kuri-Morales
Instituto Tecnológico Autónomo de México
Río Hondo No. 1, México D.F.
akuri@rhon.itam.mx

Abstract. The solution of Systems of Simultaneous Non-Linear Equations (SNLE) remains a complex and as yet not closed problem. Although analytical methods to tackle such problems do exist, they are limited in scope and, in general, demand certain prior knowledge of the functions under study. In this paper we propose a novel method to numerically solve such systems by using a Genetic Algorithm (GA). In order to show the generality of the method we first prove a theorem which allows us to equate the SNLE problem to that of minimizing a linear combination of the equations to be solved, subject to a set of constraints. Then we describe a rugged GA (the so-called Vasconcelos GA or *VGA*) which has been proved to perform optimally in a controlled but unbounded problem space. Next, we show how the VGA may be efficiently utilized to adapt its behavior (which is inherently adequate to solve unconstrained minimization problems) to a constrained solution space. Finally, we offer some examples of systems of non-linear equations which were solved using the proposed methodology.

Keywords. Simultaneous non-linear equations, minimization, genetic algorithms, constrained problems.

1 Introduction

There are several ways to attempt the numerical solution of systems of simultaneous non-linear equations [1], [2], [3]. However, the usual methods are generally lacking in that the user has to determine a solution seed or set of solutions seeds in such a way that the (generally iterative) algorithm is able to improve on those seeds until a reasonable answer is found. If, however, the user fails to properly select the seed(s) the iterations may not converge. On the other hand, closed methods demand the equations to comply with some mathematical properties which tend to invalidate the method when they are not met. Evolutionary algorithms, on the other hand, do not require that the initial iteration space is seeded. In fact, it has been proven [4] that GAs will converge to an adequate solution regardless of the initial population. Furthermore, no particular mathematical properties of the function under scrutiny are needed for the algorithm to traverse the problem space and, eventually, reach the desired global optimization (which, for simplicity and without loss of generality is normally formulated in terms of minimization). In order to be able to apply GAs to the SNLE problem we have to devise a method to transform it into a minimization one. It will be proven, in the sequel, that a simple linear combination of the equations to be simultaneously solved, subject to a, likewise, simple set of restrictions will ultimately do the trick, provided that we find a) An efficient GA and b) A scheme to handle restrictions with the purported GA.

The rest of the paper is organized as follows. In part 2 we describe the method and prove the solving capability of the linear combination of the original equations. In part 3 we discuss the VGA and how it was shown to perform in a markedly superior way relative to other types of GAs. In part 4 we show that the inherently non-constrained minimization evolutionary methods may be easily complemented so that they are able to successfully tackle constrained minimization. In part 5 we describe some sets of SNLEs which were solved with the proposed method. Finally, in part 6 we offer our conclusions.

2 SNLE Algorithm

In order to present the algorithm which solves the SNLE problem we will exemplify with a simple problem. Next we will show that the method may be generalized. The reader should keep in mind that the minimization method which is implied in what follows is a Genetic Algorithm. Therefore, we need to specify both the format in which the variables are to be represented as well as the code in which such representation is ultimately implemented. Hence, although not explicitly stated, the minimization discussed in the sequel automatically implies the explicit manipulation of the

internal (binary) representation of the variables to be considered. This is not common with other methods, where the task above is left to the compiler and, ultimately, depends on the architecture of the computer where the algorithms are implemented.

2.1 A Simple SNLE Problem

Let us solve the following set of non-linear equations:

$$\begin{aligned} &3x^2+\sin(xy)-z^2+2=0\\ &2x^3-y^2-z+3=0\\ &\sin(2x)+\cos(yz)+y-1=0 \end{aligned} \qquad (1)$$

These equations are typically representative of the kind of problems we wish to tackle: a) The system displays non-linearities in all three of the variables, b) The trigonometric functionals provide us with a potentially large number of solutions and c) The solutions are not immediate from the equations. We now proceed as follows:

1. Represent each of the variables in a binary fixed point format with an S3.25 format. That is, every variable is represented in binary assigning 1 bit to the sign, 3 bits to the integer part and 25 bits to the decimal part. This choice already implies the definition of a feasible space of solutions: in this case $-8+2^{-25} \le x, y, z \le +8-2^{-25}$. The selection of the particular number of integer and decimal bits is arbitrary but has no bearing on the algorithm, as will be shown in the sequel.
2. Add all of the equations in (1) to get the expression:

$$W = 2x^3+3x^2+sin(2x)-y^2+sin(xy)+\cos(yz)+y-z^2-z+4 \ldots \qquad (2)$$

3. Minimize W subject to the following constraints:

$$\begin{aligned} C_1 &= 3x^2+\sin(xy)-z^2+2 \ge 0\\ C_2 &= 2x^3-y^2-z+3 \ge 0\\ C_3 &= \sin(2x)+\cos(yz)+y-1 \ge 0 \end{aligned} \qquad (3)$$

By following this simple procedure, we get the following values for W, x, y and z:

W = +0.038767
x = -0.0354308933
y = +1.2499999553
z = +1.3998183608

Furthermore, the C_1, C_2 and C_3 attain the following values:

C_1= 0.00000005
C_2= 0.03759279
C_3= 0.00117488

We can see, therefore, that the algorithm is promptly lead to values numerically close to the solution (we differ the discussion of the optimization algorithm for the next section). The reason is simple: a) We bundle together all the conditions of the problem by linearly combining all the equations and b) We restrict the values of the variables in such a way that, *when minimizing*, we are lead to a lower bound which approaches the solution to the desired equations. That is, in trying to minimize the *global* expression, the algorithm tends to find the smallest values corresponding to the constraints. These last constitute the solution to the problem. In that sense, any minimization method will yield similar results, provided that we can "guarantee" its convergence.

This intuitive notion is formalized in what follows. But notice that we could have equally well stated the restrictions in (3) as follows:

$$\begin{aligned} C_1 &= 3x^2+\sin(xy)-z^2+2 \le 0\\ C_2 &= 2x^3-y^2-z+3 \le 0\\ C_3 &= \sin(2x)+\cos(yz)+y-1 \le 0 \end{aligned} \qquad (4)$$

provided we now *maximize* expression (2). Then we approach the desired values from below.

2.2 Linear Combination Theorem

It is natural to ask ourselves whether the method may be applied with generality, that is, if we can apply the method to any system of equations where a solution exists. This is proved in what follows.

Lemma.

Let $f_i(\bar{x})$ be a set of functions defined in $S_i \subset \Re^n \to \Re$ for *i=1, ..., m* and *A* a subset in $\Re$ such that $A=\{\alpha \mid \alpha = \sum_{i=1}^{m} c_i f_i(\bar{x});\ \bar{x} \subset S_i \text{ and } c_i f_i(\bar{x}) \ge 0\ \forall i\}$ where $c_i \ne 0$. If $\#A \ne 0$ and $a \in A$ then $a \ge 0$.

Proof.

Let

$$a \in \{\alpha \mid \alpha = \sum_{i=1}^{m} c_i f_i(\bar{x});\ \bar{x} \subset S_i \text{ and } c_i f_i(\bar{x}) \ge 0;\ \forall i\}$$

then there exists $(\#A \neq 0)$ an $\bar{x}_0$ such that $a = \sum_{i=1}^{m} c_i f_i(\bar{x}_0)$ and $c_i f_i(\bar{x}_0) \geq 0$ which implies that a is the summation of *m* real positive numbers. Hence, $a \geq 0$.

Theorem 2.2

Let $f_i(\bar{x}) = 0$ for *i = 1, ..., m* be a system of *m* equations for which there is, at least, an $\bar{x}_0$ which solves the system and $F(\bar{x}) = \sum_{i=1}^{m} c_i f_i(\bar{x})$. Then

$F(\bar{x}_0) = \min(A)$

Proof.

First, we shall prove that $F(\bar{x}) \in A$. It is easy to see that $\bar{x}_0 \in S_i$, since $\bar{x}_0$ is a solution to the system $f_i(\bar{x}) = 0$ and that it must also satisfy $c_i f_i(\bar{x}) \geq 0$. This last condition is seen to be true because, since $\bar{x}_0$ is a solution to the system, we have that $c_i f_i(\bar{x}_0) = c_i \times 0 = 0$ is true and, thus, $c_i f_i(\bar{x}) \geq 0$ also holds.

From the lemma, we know that $\min(A) \geq 0$ and, since $F(\bar{x}_0) \in A$ and $F(\bar{x}_0) = 0$ then $F(\bar{x}_0) = \min(A)$.

Corollary.

Let $B = \{\beta \mid \beta = F(\bar{y}) \text{ and } F(\bar{y}) = \min(A)\}$ where $\bar{y}$ is any vector, #B=1 and there exists an $\bar{x}_0$ which is a solution of $f_i(\bar{x}) = 0$ for *i=1,...,m*. Therefore, all the elements of B are zeroes (from the lemma) and each of the vectors $\bar{y}$ satisfies $F(\bar{y}) = 0$ for *i=1,...,m*. Then the system of equations has #B solutions.

Proof.

From theorem 2.2 we know that $F(\bar{x}) = \min(A)$ and $F(\bar{x}) = 0$ and, therefore, all the elements of B are zeroes (from the lemma) and each of the vectors $\bar{y}$ satisfies $F(\bar{y}) = 0$. But since $F(\bar{y})$ is the summation of values which are all greater than or equal to 0 (because $F(\bar{y}) \in A$) then $f_i(\bar{y}) = 0$ for *i=1,...,m* and there are as many solutions to the system of equations as there are elements in B.

What theorem 2.2 says then, is that when systems such as (1) have a solution we may always find it by minimizing the summation of a linear combination of the equations, where the values of the variables are properly constrained and, furthermore, that if more than one exists, we will find at least one of these. That is, given

$$\begin{aligned} f_1(x_1,...,x_m) &= 0 \\ f_2(x_1,...,x_m) &= 0 \\ &\cdots \\ f_3(x_1,...,x_m) &= 0 \end{aligned} \tag{5}$$

we may always find one of the solution vectors $\bar{x}_0$ from the following expression (we have arbitrarily selected $c_1 = c_2 = ... = c_m = 1$):

$$x_0 = \min[\sum_{i=1}^{m} f_i(\bar{x})] \mapsto f(\bar{x}) \geq 0 \tag{6}$$

where "$\mapsto$" means "subject to".

3 Genetic Algorithms

Implicit in the previous discussion is the fact that we have an adequate method to achieve the desired minimization. This is, in general, not true. However, we benefit from genetic algorithms and their known (proven properties) to satisfy our need.

Genetic Algorithms (GAs) are optimization meta-heuristics which differ from other optimization methods in two essential ways: a) They explore the solution landscape in several simultaneous loci and b) The algorithm explores the actual solution landscape but modifies its own behavior by changing a mapping of the space of encoded hypothetical solutions into the space of the actual solutions. They form part of what is now termed Evolutionary Computation. The "evolutionary" part of the name comes from the fact that the mentioned hypothetical solutions are refined step-by-step by preserving the most promising candidates from a so-called "population" which is, simply put, a set of viable solutions to the problem at hand (each solution in the set is an "individual" of the population). From a suggestive analogy with the living beings, the elements of the encoded solutions are called *genes* (hence the name "genetic"), the decoded solutions are the *phenotypes*, whilst their encoded counterparts are called the *genotypes*. GAs have been widely treated in the literature (see, for instance [5]) and we know that:
a) Elitist GAs will find the best solution given enough time, b) Such GAs may find solutions very close to the

best solution in logarithmic time and c) Some "simple" problems may lead the classical GAs astray lest some modifications are introduced [6]. Previous studies [7] have statistically proven that the Vasconcelos' variation of a GA (VGA) performs very favorably and avoids the pitfalls present in more naive variations. In this minimization problem, it is natural to use a VGA in an attempt to reach very good solutions in short time. In what follows we briefly describe the VGA.

3.1 VGA

In what follows we assume an algorithm which works with N individuals, of which a certain subset is chosen (selected) to be crossed and mutated; with a probability P_c of crossing over the members of the population and a probability P_m of mutating such members and that this process of selection-crossover-mutation is performed G times. The details of Vasconcelos' Genetic Algorithms are as follows.

1. a) $N \leftarrow 50$ (individuals in the population)
   b) $P_c \leftarrow 0.9$ (probability of crossover)
   c) $P_m \leftarrow 0.005$ (probability of mutation)
   d) $G \leftarrow 500$ (number of generations)
2. Calculate β (the number of bits to mutate) as:

$$\beta = ceiling[N \times |l| \times P_m]$$

where $|l|$ = bits in the individual's genome (throughout we assume binary encoding of the solutions).

3. *Generate* population *P(1)* randomly.
5. *For* $t \leftarrow 1$ *to* G
6. *Evaluate P(t).*
7. *Sort* the individuals in the population according to their fitness from best to worst.
8. *Retain* the best N individuals.
9. *For* $i \leftarrow 1$ *to* $N/2$
10. *Select* individuals i and N-i+1 (say A and Z).
11. *Cross* individuals A and Z with probability P_c. If A and Z are crossed, incorporate their offspring to the population.
12. *Endfor*
13. *Mutate* β bits (in the new individuals) randomly.
14. *Endfor*

□

In step 6, obviously, only the new individuals are evaluated. In step 11, crossover is *annular*. That is, individuals' chromosomes are considered not strings, but rather *rings* in the sense that the last bit in the genome is connected to the first. The size of the ring is $|l|/2$. Annular crossover is illustrated in Figure 1.

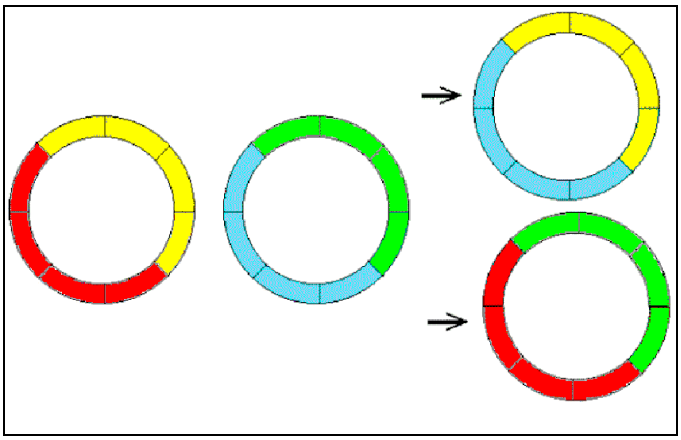

Fig. 1. Annular crossover.

As remarked, this algorithm performs very favorably and avoids the two basic recognized limitations of the so-called "simple" genetic algorithm: a) Deceptive functions and b) Spurious correlation.

4 Constrained Optimization

Constrained optimization problems are interesting because they arise naturally in engineering, science, operations research, etc. In general, a constrained numerical optimization problem is defined as:

$$\begin{aligned} &\text{Minimize} \quad f(\bar{x}) \qquad \bar{x} \in \Re^n \\ &\text{Subject to} \quad h_i(\bar{x}) = 0 \qquad i = 1,...m \\ &\qquad\qquad\quad g_i(\bar{x}) \le 0 \qquad i = m+1,...p \end{aligned} \tag{6}$$

Constraints define the feasible region, meaning that if the vector $\bar{x}$ complies with all constraints $h_i(\bar{x}) = 0$ and $g_i(\bar{x}) \le 0$ then it belongs to the feasible region. Traditional methods relying on calculus demand that the functions and constraints have very particular characteristics (continuity, differentiability, second order derivability, etc.); those based on GAs have not such limitations. For this reason, among others, it is of practical interest use the best constraint handling strategy. Here we simply discuss a method (which we call *method K*) which has been proved to perform optimally when compared with others [8]. It consists of defining the penalty function as follows: [9]

$$P(\bar{x}) = \begin{cases} \left[K - \sum_{i=1}^{s} \frac{K}{p} \right] - f(\bar{x}) & s \neq p \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where K is a large constant [$O(10^9)$], p is the number of

constraints and *s* is the number of these which have been satisfied. *K*'s only restriction is that it should be large enough to insure that any non-feasible individual is graded much more poorly than any feasible one. Here the algorithm receives information as to how many constraints have been satisfied but is not otherwise affected by the strategy. Notice that in this method the penalty is not *added* to $f(\bar{x})$ but, rather, it *replaces* $f(\bar{x})$ $\left(F(\bar{x}) = K - \sum_{i=1}^{s} \frac{K}{p}\right)$ when any of the constraints is not met. This subtle difference does, indeed, seem to make a difference (as discussed in the sequel).

4.1 Statistical Evaluation of an Algorithm

To determine the adequacy of method K we followed a statistical path. The basic reason behind this statistical methodology is to achieve the most general possible results. It is common practice to run a set of experiments to establish a comparison between proposed methods, algorithms and the like. Any such attempt is lacking since generality may not be reached from a finite set of experiments. Here we follow a similar methodology but extract hard numerical bounds because we are able to find the mean and standard deviation of the unknown distributions of all the experiments (which, for simplicity, will be referred to as *GAE*) which serve as a basis for comparison.

It is possible, indeed, to try to approximate the population's basic parameters (μ and σ) from the estimators $\bar{x}$ and s. The key issue is the adequate determination of the size of the sample. In this case we know nothing about the distribution under study (i.e. the probability that the best value from *GAE* exceeds a certain value). Hence, to find the basic parameters (which are usually calculated from $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$) we, first, rely on the fact that any sampling population of means (for a sufficiently large sample) is normally distributed. We perform enough simulations to insure that, indeed, the measured means are thusly distributed. This we ascertain by complying with a χ^2 goodness-of-fit test with a 99% level of confidence which allows us to find $\mu_{\overline{X}}$ and $\sigma_{\overline{X}}$. Second, we know that $\overline{f(\bar{x})}$ for GAE is given by $P(\overline{f(\bar{x})} < \mu_{\overline{X}} + 1.96\sigma_{\overline{X}}) \geq 0.95$ (since this distribution is gaussian). Third, for the non-gaussian distribution of $f(\bar{x})$ we further know (from Chebyshev's theorem) that $P(f(\bar{x}) < \mu_{\overline{X}} + 4\cdot[6\sigma_{\overline{X}}]) \geq 0.9375$. These bounds, on the other hand, are pertinent only for the GAE's. The point is, nonetheless, that the calculated values are absolute within statistical certainty limits and the foregoing conclusions, within these limits, are uncontestable.

4.2 The statistical algorithm

To determine the relative performance of method K, it was compared with other 4 popular methods which we denoted as methods H, J, P and S. We also selected a suite of functions and the corresponding parameters. The interested reader is referred to [8] for the details. The procedure which allows us to determine the relative goodness of any GA is as follows:

1. $\alpha \leftarrow 1$ (determine the parameter set)
   $\beta \leftarrow 1$; (determine the method)
2. $A \leftarrow M(\beta($ (where $M(i) = H, J, K, P, S$ for $i = 1, ..., 5$)
3. $i \leftarrow 1$ (count the number of samples)
4. $j \leftarrow 1$ (count the elements of a sample)
5. A function is selected randomly from the suite.
6. Experiment GAE is performed with this function and a) the best value and b) the number of satisfied constraints are stored.
7. $j \leftarrow j+1$
8. If $j \leq 36$, go to step 5 (a sample size of 36 guarantees normality).
9. The average $\bar{x}_i = \frac{1}{N}\sum_j f_j(\bar{x})$ of the best fitness' values is calculated.
10. $i \leftarrow i+1$
11. If $i \leq 50$, go to step 4
12. According to the central limit theorem, the $\bar{x}_i$ distribute normally. We, therefore, define 10 intervals which are expected to hold 1/10 of the samples assuming a normal distribution: i.e., the intervals are standardized. If the samples are indeed normally distributed the following 2 conditions should hold.

    a) At least 5 observations should be found in each of the 10 intervals (which explains why we test for 50 in step 11).

b) The values of a χ^2 goodness of fit test should be complied with (which we demand to be in the 99% confidence level).

We, therefore, check for conditions (a) and (b) above. If they have not been reached, go to step 4.

13. Once we are assured (with probability = 0.99) that the $\bar{x}_i$'s are normally distributed, we calculate the mean $\mu_{\bar{X}}$ and standard deviation $\sigma_{\bar{X}}$ of the sampling distribution of the measured mean values of the best fitnesses for this experiment. Moreover, we may calculate the mean μ and the standard deviation σ of the distribution of the best values (rather than the means) from $\mu = \mu_{\bar{X}}; \sigma = 6\sigma_{\bar{X}}$. Notice that, therefore, we characterize the statistical behavior of experiment *GAE quantitatively*.

14. $\beta \leftarrow \beta + 1$. If $\beta < 5$, go to step 3.

15. $\alpha \leftarrow \alpha + 1$. If $\alpha < 3$, go to step 2.

16. End.

□

In step 5 of the algorithm a function is randomly selected and the fitnesses of the various functions thusly chosen are averaged. However, for this to be mathematically consistent we need to normalize the results in a way such that all functions have a comparable best case. We achieve this by dividing the best measured value by the known best value (which is why we stated, above, that our choice of functions is somewhat curtailed). Therefore, the best possible (normalized) value is always Furthermore, the GAs are not guaranteed to find feasible solutions in all cases. But we must normalize even those solutions corresponding to unfeasible individuals; we did this by multiplying the number of unfulfilled constraints times the largest penalty assigned to a given individual in generation 100 of each experiment. The unmistakable result obtained from the statistical algorithm just described is that method K is superior to all the other methods tested. This is an interesting an counter-intuitive result. It seems to contradict previous works (see, for instance, [10]). Nonetheless, the results are conclusive and, because of the methodology outlined above, we are able to use method K with confidence.

5 Experiments

In what follows we describe the detailed solution of 4 SNLE problems with the VGA and method K.

5.1 Problem 1

$$\begin{aligned} x^2 + y^2 + x + y - 8 = 0 \\ xy + x + y - 5 = 0 \end{aligned} \tag{8}$$

Function to minimize:

$$f(x) = x^2 + y^2 + xy + 2x + 2y - 13 \tag{9}$$

Constraints:

$$\begin{aligned} x^2 + y^2 + x + y - 8 \geq 0 \\ xy + x + y - 5 \geq 0 \end{aligned} \tag{10}$$

This problem was solved using an S3.26 format. It was solved using two different binary encodings: a) Gray and b) Weighted binary.

The reader will recall that Gray code is cyclic and that the Hamming distance between any two consecutive codewords is 1. Intuitively, this is a nice property for any algorithm which refines the found solutions step-by-step. For it is easy to see that two numerically consecutive numbers (such as, say, "7" and "8") which are close in the numerical sense, are easier to handle by any such algorithm if they are also Hamming-wise closer. For instance, in typical weighted binary, "7" is encoded as "0111" and "8" as "1000". The Hamming distance between these two numbers is 4. In order to "find" "8" given that an algorithm has already found "7", it must change all the bits in one step and all in the adequate direction. On the other hand, Gray codes for "7" and "8" are, respectively, "0100" and "1100". The Hamming distance, here, is 1 and it is more likely that "7" "gives rise" to "8" in this sort of encoding.

Solution for Gray encoding:

$$x \approx 1.000584$$

$$y \approx 1.999661$$

Solution for weighted binary encoding:

$$x \approx 2.003467$$

$$y \approx 0.998059$$

Notice that the values for *x* and *y* were reversed in both cases. The true values being *x=1* and *y=2* (or *y=1* and *x=2*). The VGA ran for 50 generations and *N*=50 in each case. It is simple to see that with a binary representation of 60 bits, there are 2^{60} possible values, hence, the number of possible solutions is, roughly, $1{,}152{,}921 \times 10^{12}$. To find the solution the GA explored 50X50=2,500 values; approximately 2.2×10^{-13} % of the total; a remarkably efficient performance.

5.2 Problem 2

$$\begin{aligned} 3x^2 + 2\sin(x)\cos(y) - 10.2 &= 0 \\ 2y^2 + \cos(x)\sin(2y) - 17.3 &= 0 \end{aligned} \quad (10)$$

Function to minimize:

$$f(\bar{x}) = 3x^2 + 2\sin(x)\cos(y) + 2y^2 + \ldots \\ \cos(x)\sin(2y) - 27.5 = 0 \quad (11)$$

Constraints:

$$\begin{aligned} 3x^2 + 2\sin(x)\cos(y) - 10.2 &\geq 0 \\ 2y^2 + \cos(x)\sin(2y) - 17.3 &\geq 0 \end{aligned} \quad (12)$$

This problem was solved using an S3.26 format. It was solved using two different binary encodings: a) Gray and b) Weighted binary.

Solution for Gray encoding:

$$x \approx 1.998176$$
$$y \approx 1.2.926398$$

Solution for weighted binary encoding:

$$x \approx 2.000000$$
$$y \approx 2.929688$$

The errors were, respectively, $\varepsilon_{\text{Gray}} = 0.0005$ and $\varepsilon_{WB} = 0.594$. The VGA ran for 100 generations and N=50 in each case. Efficiency is analogous to the problem previously considered.

5.3 Problem 3

$$\begin{aligned} x^2 + 3y - z^3 + 10 &= 0 \\ x^2 + y^2 + z^2 - 6 &= 0 \\ x^3 - y^2 + z - 2 &= 0 \end{aligned} \quad (13)$$

Function to minimize:

$$f(x,y,z) = 2x^2 + x^3 + 3y + z + z^2 - z^3 + 2 \quad (14)$$

Constraints:

$$\begin{aligned} x^2 + 3y - z^3 + 10 &\geq 0 \\ x^2 + y^2 + z^2 - 6 &\geq 0 \\ x^3 - y^2 + z - 2 &\geq 0 \end{aligned} \quad (15)$$

This problem was solved using an S4.24 format. It was solved using weighted binary encoding.

Solution for weighted binary encoding:

$$x = +1.0169043839$$
$$y = -1.0210746527$$
$$z = +1.9970520139$$

and

$$f(+1.0169, -1.02107 + 1.99705) \approx 0.077133$$

The VGA ran for 500 generations and N=50. Here the binary encoding of 87 bits (2^{87}) yields close to $154{,}742 \times 10^{21}$ possible solutions. To find the closest the GA explored 500X50=25,000 values; approximately 1.615×10^{-19} % of the total; again, a remarkably efficient performance.

The values reached for each of the 3 constraints (say c_1, c_2 and c_3) were:

$$c_1 \approx 0.006194$$
$$c_2 \approx 0.064904$$
$$c_3 \approx 0.006033$$

5.4 Problem 4

$$\begin{aligned} 3x^2 + \sin(xy) - z^2 + 2 &= 0 \\ 2x^3 - y^2 - z + 3 &= 0 \\ \sin(2x) + \cos(yz) + y - 1 &= 0 \end{aligned} \quad (16)$$

Function to minimize:

$$f(x,y,z) = 2x^3 + 3x^2 + \sin(xy) - y^2 + \ldots \\ \cos(yz) + y - z^2 - z + 4 \quad (17)$$

Constraints:

$$\begin{aligned} 3x^2 + \sin(xy) - z^2 + 2 &\geq 0 \\ 2x^3 - y^2 - z + 3 &\geq 0 \\ \sin(2x) + \cos(yz) + y - 1 &\geq 0 \end{aligned} \quad (18)$$

This problem was solved using an S3.25 format. It was solved using weighted binary encoding.

Solution for weighted binary encoding:

$$x = -0.0391670614$$
$$y = +1.2728857100$$
$$z = +1.3738022745$$

and

$$f(x,y,z) \approx 0.090945$$

The VGA ran for 500 generations and N=50. As above, the binary encoding of 87 bits (2^{87}) yields close to $154{,}742 \times 10^{21}$ possible solutions; 500X50=25,000 values were explored; approximately 1.615×10^{-19} % of the total..

The values reached for each of the 3 constraints (say c_1, c_2 and c_3) were:

$$c_1 \approx 0.0002805$$
$$c_2 \approx 0.0037991$$
$$c_3 \approx 0.0000287$$

6 Conclusions

We have shown that it is possible to solve arbitrary sets of non-linear equations using a GA. We have proved that the results we have illustrated are applicable in general. From previous analysis we have, furthermore, provided a general optimization technique and a method (method K) which allows the GA to perform properly even when faced with constrained optimization problems.

All the foregoing conclusions apply without prior knowledge of the equations' behavior. And we have also illustrated two interesting facts: a) That binary fixed point encoding is adequate for our purposes and that b) The method is impervious to the particular type of codification.

It is possible to extend the method to include the numerical solution of the following problems: 1) Finding the roots of higher degree equations and 2) Solving Diophantine equations.

Since GAs are powerful optimization tools, it is to be expected that their application to numerically complex problems continues to expand. Particularly when there are no known methods to tackle the problems in question or when the known methods are inefficient or incomplete.

References:

[1] Roos, B. and Widmark, P. Eds. *European Summerschool in Quantum Chemistry, Book I* , http://www.teokem.lu.se/esqc/book/content1.html

[2] Systems of Non-Linear Equations, http://www.math.neu.edu/undergrad/calc4eng/q6/systems_review.html.

[3] Solving systems with linear and non-linear terms, http://www-sop.inria.fr/saga/logiciels/ALIAS/node17.html

[4] Rudolf, G., "Convergence Analysis of Canonical Genetic Algorithms", IEEE Transactions on Neural Networks, **5**(1):96-101, January, 1994.

[5] Back, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[6] Kuri, A., *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning. Theory and Applications*, Vol. 1. Instituto Politécnico Nacional, pp 270, 1999.

[7] Kuri, A., "A Methodology for the Statistical Characterization of Genetic Algorithms", Lecture Notes in Artificial Intelligence, LNAI 2313, Springer-Verlag, pp. 79-88, 2002.

[8] Kuri, A., and Gutiérrez, Jesús, "Penalty Function Methods for Constrained Optimization with Genetic Algorithms: A Statistical Analysis", Lecture Notes in Artificial Intelligence, LNAI 2313, Springer-Verlag, pp. 108-117, 2002.

[9] Kuri, A., "A Universal Eclectic Genetic Algorithm for Constrained Optimization", 1998, *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98*, pp. 518-522.

[10] Coello, C., "Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems", Computers in Industry, 41(2):113-127, 2000.