Population Induced Instabilities in Genetic Algorithms for Constrained Optimization

Population Induced Instabilities in Genetic Algorithms for Constrained Optimization

D.S. Vlachos and K.J. Parousis-Orthodoxou

Department of Computer Science and Technology,
Faculty of Sciences and Technology,
University of Peloponnese,
GR-221 00 Tripolis, Greece

E-mail: `dvlachos@uop.gr`

Abstract. Evolutionary computation techniques, like genetic algorithms, have received a lot of attention as optimization techniques but, although they exhibit a very promising potential in curing the problem, they have not produced a significant breakthrough in the area of systematic treatment of constraints. There are two mainly ways of handling the constraints: the first is to produce an infeasibility measure and add it to the general cost function (the well known penalty methods) and the other is to modify the mutation and crossover operation in a way that they only produce feasible members. Both methods have their drawbacks and are strongly correlated to the problem that they are applied. In this work, we propose a different treatment of the constraints: we induce instabilities in the evolving population, in a way that infeasible solution cannot survive as they are. Preliminary results are presented in a set of well known from the literature constrained optimization problems.

1. Introduction

During the last years, genetic algorithms have received much attention regarding their potential as global optimization techniques. Since one of the most interesting and challenging field in global optimization is the class of constrained problems (both linear and nonlinear), a lot of effort has been recorded from researcher in this direction and several techniques have been developed and embedded in standard genetic algorithms ([1], [2], [3], [4], [5], [6], [7]). A nice short review of proposed methods with selected references can be found in [8].

In general, there are two ways in handling the constraints. The first one is the penalty function methods, in which a penalty term is added to the objective function. This term should be zero in the case of feasible solutions while in the case of constrains violation, the penalty increases. The second method is to take extra actions in order to limit the members of the population in the feasible region of the search space. There are several techniques that have been proposed, from modified mutation and crossover operation with the property that they only produce feasible offsprings from feasible parents to the death penalty method in which a member is destroyed if it violates a certain constrain.

Both methods have their pros and cons. Penalty methods are simple and self-evident. On the other hand, as far as the second approach is concerned, searching in the feasible region of the search space can be a very complicated task. The method is equivalent to transform the constrained problem to a constrain-free one. In any case, the basic requirement is that infeasible members cannot survive for long. It is important that the method allows for infeasible members in order to preserve the searching capabilities of the genetic algorithm, but sooner or later, these members must be replaced by feasible ones.

this work we develop a new approach to obtain this. Assuming a binary coding of the chromosomes, the representation of the population of the genetic algorithm can be viewed as an array of binary digits that oscillate with time (generations). As the solution is approached, these oscillations are decreased until they are disappear when the genetic algorithm reaches the global minimum. Instead of replacing infeasible solutions with feasible ones (modifying and thus oscillate the binary digits), we could obtain the same result by inducing instabilities (oscillations) to the infeasible members of the populations without modifying the objective function. In the following, we shall investigate this idea by tuning the mutation and crossover probabilities according to the feasibility of a member of the population.

2. Formulation

Consider the general nonlinear programming problem

$$Minimize\ f(x),\ x \in R^n \tag{1}$$

subject to q equality constraints

$$h^i(x) = 0\ ,\ i = 1...q \tag{2}$$

and $m - q$ inequality constraints

$$g^j(x) \geq 0\ ,\ j = q+1...m \tag{3}$$

We can now define a measure of infeasibility c^i of constrain i by

$$c^i(x) = w^i \cdot \begin{cases} \delta_b(h^i(x)) & ,\quad i = 1...q \\ u_a(g^i(x)) & ,\quad i = q+1...m \end{cases} \tag{4}$$

where w^i is a weight in order to bring the constrains in a comparable between them range of values and a, b are positive constants. The functions u_a and δ_b are given:

$$u_a(x) = \frac{\lambda(ax)}{1 - \lambda(ax)} \tag{5}$$

and

$$\delta_b(x) = \frac{\lambda(1 - b|x|)}{1 - \lambda(1 - b|x|)} \tag{6}$$

where $\lambda(x)$ is a continuous and at least k-times differentiable function (the value of k will be calculated later) with the property that is zero if $x \geq 0$, and for $x < 0$ is increasing and

$$\lim_{x \to -\infty} \lambda(x) = 1 \tag{7}$$

It can be easily proved now that $c^i(x)$ equals to zero for feasible solutions and $c^i(x)$ are everywhere C^k functions if this is true for the functions $h^i(x)$ and $g^j(x)$. Moreover, a very important property of $u_a(x)$ and $\delta_b(x)$ is that their n-th derivative at $x = 0$ is zero for every $n \leq k$. Fig. 1 shows the general shape of the functions $u_a(x)$ and $\delta_b(x)$ for different values of a and b. The values of the functions $c^i(x)$ for a specific member x of the population, measure how

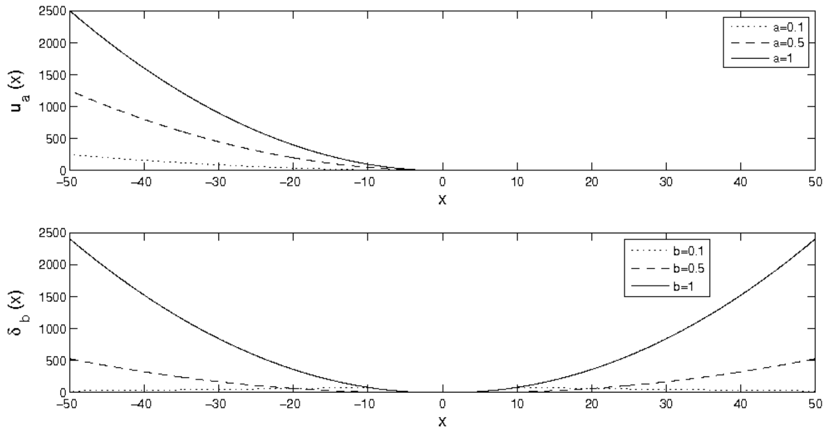

Figure 1. Functions $u_a(x)$ and $\delta_b(x)$ for different values of a and b.

deep in the infeasible region, the member lies. Consider now a feasible member x. If we induce a change Δx in the member, the infeasible measure becomes

$$c^i(x + \Delta x) = \frac{\partial^{k+1} c^i}{\partial x^{j_1} ... \partial x^{j_{k+1}}} \Delta x^{j_1} \cdot ... \cdot \Delta x^{j_{k+1}} + \mathcal{O}(|\Delta x|^{k+2}) \tag{8}$$

while for an infeasible member, the infeasibility measure becomes

$$c^i(x + \Delta x) = c^i(x) + \frac{\partial c^i}{\partial x^j} \Delta x^j + \mathcal{O}(|\Delta x|^2) \tag{9}$$

The vector $\frac{\partial c^i}{\partial x^j}$ now shows us the direction of the steepest descent (or ascent) of the infeasibility measure c^i. An important conclusion of the above analysis, is that, if we let k to be big enough, then any small change in a feasible member, will produce a feasible member again, since from eq. (8) the infeasibility measure will be negligible. The ideal case is if we can make $k \to \infty$. Fortunately, this can be obtained if we select

$$\lambda(x) = \begin{cases} e^{-\frac{1}{x^2}} & , \quad x < 0 \\ 0 & , \quad x \geq 0 \end{cases} \tag{10}$$

It can be easily proved that $\lambda(x)$ is a $\mathcal{C}^\infty$ function with $\lambda^{(n)}(x) = 0$ for every n. Using now eq. (10) to define the infeasibility measure, we have ensured that (i) every infeasible member will have a measure which increases with increasing distance of the member from the feasible part of the search space, (ii) every feasible member will have a null measure and most important that (iii) a small change of a feasible member will produce again a feasible one. We can now use these properties and define mutation and crossover probabilities in order to induce instabilities in the infeasible members.

2.1. Mutation
For every member x of the population, we construct the vector s_j by

$$s_j = \sum_{i=1}^{m} \frac{\partial c^i}{\partial x^j} \tag{11}$$

If the $|s|$ equals to zero, then we use the standard mutation probability, otherwise we produce a random integer number j between 1 and m with probability

$$p(j) = \frac{s_j}{|s|} \tag{12}$$

and mutate a random bit in the binary encoding of the j-th part of the chromosome representing the member x. Here we assume that the chromosome uses a Gray binary code to encode the n-variables of the problem. Each variable occupies a continuous part of bits in the chromosome.

The result of this operation is that every infeasible member will mutate, making impossible the stabilization of the solution in the infeasible area of the search space. But for a feasible member, this mechanism produce no effect. Mutation will occur as in the standard genetic algorithm in order to span the search space.

2.2. Crossover

Consider now the case where a member x of the population is candidate for crossover. There are two possibilities. The first one is that x is a feasible solution. In this case we should enhance crossover with the hope that offsprings from x will inherit the feasibility. The other case is when x is an infeasible solution. Suppose that the other parent is member y. If y is a feasible solution, then again we use the standard crossover probability. If y is infeasible, then it might violate the same or different constraints as x. If the same constraints are violated from both x and y, then it is likely that offsprings will inherit this property. But if x and y violate different constraints, it is possible that an offspring inherits the good part of the genetic information from each parent. In order to quantify this idea, we modify the standard crossover probability p_c by

$$p'_c = p_c/(1 + s_x \cdot s_y) \tag{13}$$

where s_x and s_y are the vectors defined in eq. (11) for the parents x and y. It is clear that if any of the members is feasible, then the crossover probability is that of the standard genetic algorithm, but if both x and y are infeasible, the crossover probability is decreased with increasing number of common violated constrains.

2.3. Selection

The selection mechanism is important for two reasons: first it enhances the best individuals to crossover and second, it selects between the individuals the best ones to survive. Since the searching mechanism of a genetic algorithm is based on crossover and mutation, the selection mechanism forces the searching to be performed in areas of the search space which are close to the best individuals. By enhancing crossover of the best individuals, the selection mechanism results in searching near these individuals and by allowing them to survive, this searching is continued. Selection can be performed only by defining an ordering relation between the members of the population, and this is achieved by the fitness function and in the case of constrained optimization by both the fitness and penalty functions.

Consider now a different mechanism which leads to the same result as the selection mechanism described before. Every member of the population has its own clock which counts how much time this member has lived. A member dies if the number of ticks of its clock cross a predefined level. The frequency of the clock is proportional to the value of the objective function. Thus, the smaller the value of the objective function for a member, the slower the frequency of the clock of this member and the longer this member will live. Let

$$F = f(x) \; and \; C = \sum_{i=1}^{m} c^i(x) \tag{14}$$

be the value of the objective function and a measure of the infeasibility for a given member x respectively. Now, combine these two measures to create the clock frequency of the member

$$\Omega = F + iC \tag{15}$$

The member x continues to survive as long as

$$\frac{|e^{i\Omega t}|}{winding\ number\ of\ curve\ e^{i\Omega t}} > threshold \tag{16}$$

where the winding number of a closed curve in the plane around a given point is an integer representing the total number of times that curve travels counterclockwise around the point and *threshold* is a predefined value. For a feasible member, the numerator of equation (16) is always 1 and that member dies when

$$\frac{F \cdot t}{2\pi} > \frac{1}{threshold} \tag{17}$$

The condition now that best members live more (and thus crossover and mutate more) is fulfilled. On the other hand, for an infeasible member, the numerator of equation (16) decreases with time and thus this member will die sooner.

2.4. Technical details

It is clear from the analysis of the proposed algorithm that the number of the individuals in the population is variable. In the first few generations, every member of the population survives, so it is a good practice to start with a small size of the population. Since the value of the objective function may change for a specific member (due to mutation), we have to update at every generation and for all members the phase Ωt. Finally, it is a natural selection to transform every calculated quantity in the population according to its average value. Since evolution operators depend on probabilities, only relevant probabilities are of interest.

3. Experimental results

The performance of the proposed constraint handling method has been evaluated using a set of 11 test cases ([9],[10]). These test cases include various forms of objective function (linear, quadratic, cubic, polynomial, nonlinear), and each test case has a different number of variables (n). The test problems also pose a range of constraint types and number of constraints [linear inequalities (LI); nonlinear equalities (NE); and nonlinear inequalities (NI)]. The general form of each test case is given in Table 1, which also indicates the number of constraints active at the optimum solution (a). Note here that every optimization problem is transformed in the form of eq. (1). For example, in $G3$ instead of maximizing $\sqrt(n)^n \prod_{i=1}^n x_i$, we minimize the function $1 - \sqrt(n)^n \prod_{i=1}^n x_i$.

Fig. 2 shows the result of the application of the new method in problems $G1$ to $G11$. The error is the distance of the solution calculated from the best known solution. In this error, we have to include the quantization error, since for the needs of this experiment we use an 8-bit encoding of each variable, so for variable i we have an extra error e_i given

$$e_i = \frac{range\ of\ variable\ _i}{256} \tag{18}$$

The starting size of the population in all test problems was 50 members and this size changed during the experiment as it was explained before. The probabilities for standard crossover and mutation was 1 and 0.06 respectively. The horizontal axis of Fig. 2 corresponds to the number of generations and of course this is in a one to one correspondence with the computational time

Case	**n**	**Objective**	**LI**	**NE**	**NI**	**a**
G1	13	quadratic	9	0	0	6
G2	k	nonlinear	0	0	6	1
G3	k	polynomial	0	1	0	1
G4	5	quadratic	0	0	6	2
G5	4	cubic	2	3	0	3
G6	2	cubic	0	0	2	2
G7	10	quadratic	3	0	5	6
G8	2	nonlinear	0	0	2	0
G9	7	polynomial	0	0	4	2
G10	8	linear	3	0	3	6
G11	2	quadratic	0	1	0	1

Table 1: Summary of Test Cases

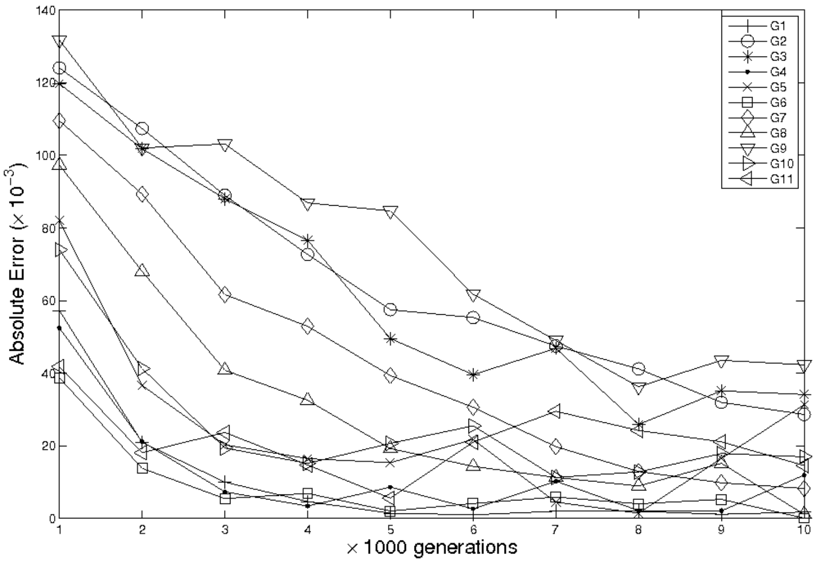

Figure 2. Absolute error vs number of generations for the eleven problems of table 1. In problems $G2$ and $G3$ the number of the unknown variables was 10.

of the method. The method seems to behave well in the case of linear constraints, although it exhibits a slow convergence in the case of non linear ones, especially $G2, G3$ and $G9$. On the other hand, the method treats successfully the case of boundary searching ($G1, G7$ and $G10$ for example).

Finally, it is interesting to see the dynamics of the method. First, the population size increases during the first generations. But later, there is an almost stable population size because the number of new offsprings are equal to the number of members that cannot survive. This is shown if Fig. 3a and for problem $G1$. On the other hand, the frequency of the bit changes in the binary representation of members is shown in Fig. 3b for the feasible and infeasible members and

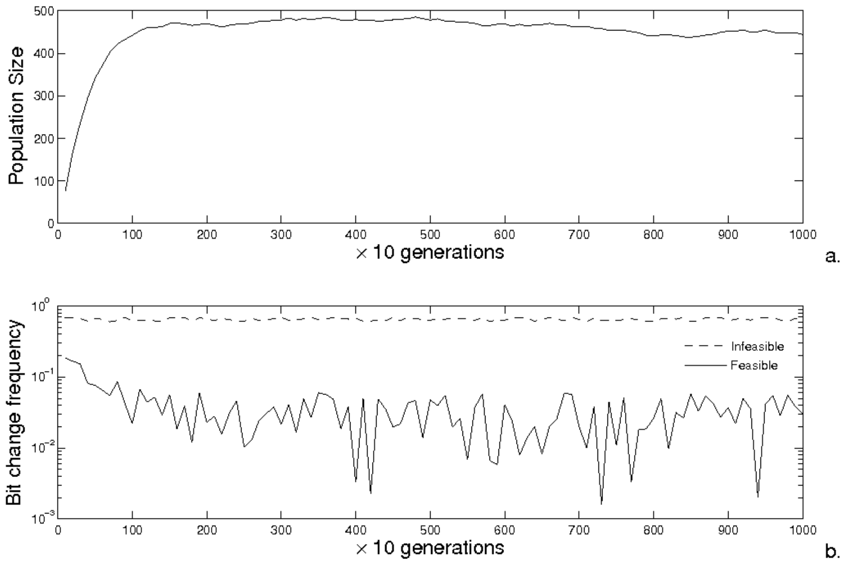

Figure 3. (a) The population size vs the number of generations. (b) The frequency of bit changes in the binary representation of infeasible (dashed line) and feasible (solid line) members vs the number of generations.

for the same problem $G1$. It is clear that bits in infeasible members oscillate with a frequency which is one order of magnitude greater than the bit oscillation caused by standard mutation and crossover.

4. Conclusion

A new approach for handling constrains in optimization using genetic algorithms is presented in this letter. The new method induces instabilities in the infeasible members of the population by mutation and selective crossover, while it introduces a new mechanism for selection by letting every member of the population to survive a number of generations which depends on both the fitness and the feasibility of the member. The preliminary results from the application of the method to a well known academic set of constrained problems is very promising. Since the method is parameter free, it might be used for complex constrained optimization problems.

References

[1] Hamida S B and Schoenauer M 2000 *Proc. Parallel Problem Solving from Nature* **VI** 5296–538
[2] Coit D W, Smith A E and Tate D M 1996 *INFORMS J. Comput.* **8** 1736–182
[3] Deb K 2000 *Computer Methods in Applied Mechanics and Engineering* **186** 3116–338
[4] Hadj-Alouane A B and Bean J C 1997 *Oper. Res.* **45 (1)** 926–101
[5] Powell D and Skolnick M M 1993 *Proceedings of 5th Int. Conf. Genetic Algorithms*
[6] Runarsson T P and Yao X 2000 *IEEE Trans. Evol. Comput.* **4** 2846–294
[7] Wright J A and Farmani R 2001 *Proc. Genetic and Evolutionary Computation Conf.* (San Francisco, CA) pp 7256–732
[8] Farmani R and Wright J A 2003 *IEEE Trans. Evol. Comput.* **7** 4456–455
[9] Michalewicz Z and Fogel D B 2000 *How to Solve It: Modern Heuristics* (Berlin, Germany: Springer-Verlag)
[10] Koziel S and Michalewicz Z 1999 *Evolutionary Computation* **7 (1)** 196–44