Penalty Function Methods for Constrained Optimization with Genetic Algorithms: A Statistical Analysis

Angel Fernando Kuri-Morales[1] and Jesús Gutiérrez-García[2]

[1] Instituto Tecnológico Autónomo de México
Río Hondo No. 1, México D.F.
akuri@rhon.itam.mx
[2] Centro de Investigación en Computación
Instituto Politécnico Nacional, México D.F.
jgg@cic.ipn.mx

Abstract. Genetic algorithms (GAs) have been successfully applied to numerical optimization problems. Since GAs are usually designed for unconstrained optimization, they have to be adapted to tackle the constrained cases, i.e. those in which not all representable solutions are valid. In this work we experimentally compare 5 ways to attain such adaptation. Our analysis relies on the usual method of selecting an arbitrary suite of test functions (25 of these) albeit applying a methodology which allows us to determine which method is better within statistical certainty limits. In order to do this we have selected 5 penalty function strategies; for each of these we have further selected 3 particular GAs. The behavior of each strategy and the associated GAs is then established by extensively sampling the function suite and finding the worst case best values from Chebyshev's theorem. We have found some counter-intuitive results which we discuss and try to explain.

1 Introduction

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:

$$\begin{array}{lll} \text{Minimize} & f(\vec{x}) & \vec{x} \in \Re^n \\ \text{Subject to} & h_i(\vec{x}) = 0 & i = 1,\ldots m \\ & g_i(\vec{x}) \le 0 & i = m+1,\ldots p \end{array} \tag{1}$$

Without loss of generality we may transform any optimization problem to one of minimization and we, therefore, develop our discussion in such terms. Constraints define the feasible region, meaning that if the vector $\vec{x}$ complies with all constraints $h_i(\vec{x}) = 0$ and $g_i(\vec{x}) \le 0$ 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 to be able to ascertain which of the many proposed constraint handling strategies is best.

© Springer-Verlag Berlin Heidelberg 2002

This paper is organized in 4 further sections. Section 2 succinctly describes the methods under analysis; in section 3 we describe the experiments performed; in sections 4 and 5, finally, we present our results and conclusions.

2 Strategies

The strategies we selected are variations of what is the most popular approach to constrained optimization: the application of penalty functions [1]. In this approach, a constrained problem is transformed into a non-constrained one. The function under consideration is transformed as follows:

$$F(\vec{x}) = \begin{cases} f(\vec{x}) & \vec{x} \in \textit{feasible region} \\ f(\vec{x}) + \textit{penalty}(\vec{x}) & \vec{x} \notin \textit{feasible region} \end{cases} \tag{2}$$

and the problem described in (1) turns into the one of minimizing (2) if a proper selection of the penalty function is achieved. We now describe the way in which this penalty function (denoted by $P(\vec{x})$) has been tackled in the strategies we selected. In what follows we denote Homaiffar's, Joines & Houck's, Schoenauer & Xanthaki's, Powell & Skolnick's and Kuri's methods as methods H, J, S, P and K, respectively.

2.1 Method H

This strategy was originally described in [2]. It defines l penalty levels depending on the magnitude of the violation of the constraints. To define such levels it demands to define intervals for each of the violations and a penalty value for every interval.

$$P(\vec{x}) = \begin{cases} 0 & \vec{x} \in M \\ \sum_{i=1}^{m} R_{i,j} H_i^2(\vec{x}) & \vec{x} \notin M \end{cases} \tag{3}$$

M is the set all feasible individuals; index *i* refers both to constraints of inequality and equations (g_i and h_i respectively); the function H is defined as the maximum value between 0 and g_i for *i=1,...m* and the absolute value of h for $i = m+1, ..., p$. Constant *R* is defined as follows:

$$R_{ij} = \begin{cases} R_{i,1} & \text{if } a_{0,i} < H_i(\vec{x}) < a_{1,i} \\ R_{i,2} & \text{if } a_{1,i} < H_i(\vec{x}) < a_{2,i} \\ & \quad\cdots \\ R_{i,l} & \text{if } a_{l-1,i} < H_i(\vec{x}) < a_{l,i} \end{cases} \tag{4}$$

This method requires the definition of *m(2l+1)* parameters which remain constant throughout. Hence, this is a *static penalty method.*

In our experiments it was impossible to consider special values for R_{ij} in every function and, hence, we decided to utilize 4 penalty levels with R = 100, 200, 500, 1000 (instead of 50, 60 and 90 as reported in [1]) and intervals of (0-10), (10-100), (100-1000) and (1000-∞).

2.2 Method J

The original description of this method may be found in [3]. In it a dynamic (non-stationary) penalty function is defined. That is, the penalty, function changes as the GA proceeds. The definition is as follows:

$$P(\vec{x},\alpha,\beta)=\rho_k^{\alpha}\times SVC(\beta,\vec{x}) \tag{5}$$
$$\rho_k = C\times k \quad k=\#\,generation$$
$$SVC(\beta,\vec{x})=\sum_{i=1}^{p} f_i^{\beta}(\vec{x}) \quad \beta=1,2,...$$

where α, β and C are parameters of the method and k is the number of generation under consideration. The values we used to test the method were C = 0.5, α = 2 and β = 2.

2.3 Method S

Shoenauer and Xanthakis's method, originally described in [4], does not only define a penalty function; it resorts to an algorithm to find feasible individuals from the evaluation of the constraints as fitness functions and eliminating those individuals which do not comply with the constraints. The algorithm is as follows:

- Start with a random population (which, in general, holds both feasible and unfeasible individuals)
- Set j = *1* (j is a constraint counter)
- Evolve this population to minimize the violation to the *j-th* constraint until a percentage (ϕ) of the population is feasible for this constraint. Then:

$$F(\vec{x})=g_j(\vec{x}) \tag{6}$$

- $j \leftarrow j+1$
- The present population is the starting point for the next phase of evolution, which consists of the minimization of the *j-th* constraint. During this phase those points which do not comply with the 1, 2, ..., *j-th* constraint are eliminated from the population.
- Stop if a percentage (ϕ) of the population which complies with the *j-th* constraint is reached.
- If $j<m$, the last two steps are repeated, otherwise (j = m) function f is minimized, rejecting all unfeasible individuals.

To implement this method it was deemed necessary to establish full elitism [5] since, otherwise, the number of individuals in the population decreased importantly for every new generation. In keeping the best *N* individuals (where *N* denotes the size of the population) we guarantee that even if there is no new individual satisfying the *j-th* constraint we can always count with all individuals of the previous generation. This insures that the size of the population remains constant even when eliminating those individuals which violate the constraints.

2.4 Method P

This method was developed circa 1993 [6] and includes a heuristic to single out non-feasible points: "Any feasible solution is better than a non-feasible one". The penalty function is defined as follows:

$$P(\vec{x}) = r\sum_{j=1}^{p} f_j(\vec{x}) + \rho(\vec{x},t) \qquad \vec{x} \notin M \tag{7}$$

$$\rho(\vec{x},t) = max\left\{0, \underset{x\in M}{max}\{f(\vec{x})\} - \underset{x\notin M}{min}\{f(\vec{x})\} + \sum_{j=1}^{p} f_j(\vec{x})\}\right\}$$

where *r* is a constant. The value set for *r* in our experiments is 2.

2.5 Method K

This is the simplest of all the methods considered and it consists of defining the penalty function as follows: [7]

$$P(\vec{x}) = \begin{cases} \left[K - \sum_{i=1}^{s} \frac{K}{p}\right] - f(\vec{x}) & s \neq p \\ 0 & otherwise \end{cases} \tag{8}$$

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, however, that in this method the penalty is not *added* to $f(\vec{x})$ as in (2) but, rather, it *replaces* $f(\vec{x})$ $\left(F(\vec{x}) = K - \sum_{i=1}^{s} \frac{K}{p}\right)$ when any of the constraints is not met. This subtle difference does, indeed, seem to make a difference (as discussed in the sequel).

3 Experiments

We designed three different algorithms to test the behavior of each of the methods. The intent is to explore whether the particular GA determines significant performance differences. For instance, see if Homaiffar's method is better than Joines' for a given algorithm but with a different one the opposite is true.

All individuals were Gray encoded with a fixed point format with 14 bits for the integer part and 20 bits for the decimal part, thus yielding an effective range of $-2^{15} < x < +2^{15}$ for any number *x*. The algorithms for the experiments were set as follows ($p_c \equiv$ crossover probability; $p_m \equiv$ mutation probability; $N \equiv$ population size; $\mathrm{G} \equiv$ number of generations):

G1: p_c = 0.9, p_m = 0.05, N = 100, G = 100; proportional selection; 1-point crossover; the best individual was preserved (simple elitism).

G2: p_c = 0.9, p_m = 0.07, N = 50, G = 100; proportional selection; 1-point crossover; the best 25 individuals were preserved.

G3: p_c = 0.9, p_m = 0.07, N = 50, G = 100; deterministic (Vasconcelos' [5]) selection; 1-point crossover; the best N individuals were preserved (full elitism).

The combinations in G1, G2 and G3 are arbitrary and, indeed, reflect no intention on our part but to determine whether there is a qualitative difference in the methods such that when applied to a problem with different algorithm *the method* (as opposed to the algorithm) yields significantly different results. In the analysis that follows we emphasize this goal.

To analyze the behavior of every method we selected a suite of 25 functions, (which we cannot discuss because of space), but they range from the relatively simple to solve to the very difficult to solve. Our method, as described in what follows, requires the knowledge of the solution to the problem and we were, in that sense, somewhat limited in our choice of functions. Thereafter, we executed algorithms G1, G2 and G3 and measured their behavior. Every method was tried exhaustively for every algorithm. We refer to the corresponding experiments as $G\alpha A$: for instance, G1H refers to the using algorithm G1 and Homaiffar's method; G2P singles out algorithm G2 and Powell's method. We use the letters H, J, K, P or S to refer to the different methods. To find a quantitative measure of relative performance we follow the next procedure.

3.1 Statistical Evaluation of an Algorithm

It is common 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 $G\alpha A$ experiments.

3.1.1 Extracting the Distribution's Parameters

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 $G\alpha A$ 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 $\mu_{\bar{X}}$ and $\sigma_{\bar{X}}$. Second, we know that $\overline{f(\vec{x})}$ for $G\alpha A$ is given by $P(\overline{f(\vec{x})} < \mu_{\bar{X}} + 1.96\sigma_{\bar{X}}) \geq 0.95$ (since this distribution is gaussian). Third, for the non-gaussian distribution of $f(\vec{x})$ we further know (from Chebyshev's theorem) that $P(f(\vec{x}) < \mu_{\bar{X}} + 4\cdot[6\sigma_{\bar{X}}]) \geq 0.9375$. These bounds, on the other hand, are pertinent only for the $G\alpha A$'s. The point is, nonetheless, that the calculated values are absolute within statistical certainty limits and the foregoing conclusions, within these limits, are uncontestable.

3.1.2 The Statistical Algorithm

1. $\alpha \leftarrow 1$ (determine the parameter set)
2. $\beta \leftarrow 1$; (determine the method)
   $A \leftarrow M(\beta)$ $(where\ M(i) = H, J, K, P, S\ \ for\ i = 1, ..., 5)$
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 $G\alpha A$ 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 $\bar{x}_i = \frac{1}{N}\sum_j f_j(\vec{x})$ 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 $\mu = \mu_{\overline{X}}$; $\sigma = 6\sigma_{\overline{X}}$. Notice that, therefore, we characterize the statistical behavior of experiment $G\alpha A$ *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 1. 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. This explains the large values reported for $\mu_{\overline{X}}$ and $\sigma_{\overline{X}}$ in the tables which follow.

4 Results

In table 1 we show the sampling of means average values and standard deviations for each tested method. We point out that Schoenauer's method was only simulated with G3 because it requires full elitism.

Table 1. Values of $\mu_{\overline{x}}, \sigma_{\overline{x}}$ for the experiments performed

Experiment	$\mu_{\overline{x}}$	$\sigma_{\overline{x}}$
G1K	1.47155E281	2.20060E280
G1H	2.27267E281	3.54317E280
G1J	2.32140E281	2.35599E280
G1P	2.13868E281	2.22789E280
G2K	1.66157E281	2.05705E280
G2H	2.40939E281	2.14969E280
G2J	2.29502E281	2.41732E280
G2P	2.07283E281	2.34106E280
G3K	1.61322E281	2.00831E280
G3H	2.37554E281	2.58420E280
G3J	2.14620E281	2.59664E280
G3P	1.90579E281	2.39071E280
G3S	1.92934E281	2.40484E280

Since, from Chebyshev's theorem, we know that the proportion of any distribution found within k standard deviations of the mean is, at least, $1-1/k^2$ (k is any positive number greater than 1) we decided to set *k=4* and, therefore, a probability certainty of 0.9375. Then we calculated the upper bound (worst case minimum value) for the experiments. These are shown in table 2, where the $G\alpha A$ are ordered according to this bound, from best to worst.

Table 2. Upper Bound for Best Values of $G\alpha A$ with P=0.975 (/E281)

Experiment	$\mu_{\overline{X}}$	$\sigma_{\overline{X}}$	Upper Bound	Relative Performance
G3K	1.6132	0.2008	6.4332	1.0000
G2K	1.6616	0.2057	6.5985	1.0257
G1K	1.4716	0.2201	6.7530	1.0497
G1P	2.1387	0.2228	7.4856	1.1636
G2H	2.4094	0.2150	7.5686	1.1765
G3P	1.9058	0.2391	7.6435	1.1881
G2P	2.0728	0.2341	7.6914	1.1956
G3S	1.9293	0.2405	7.7010	1.1971
G1J	2.3214	0.2356	7.9758	1.2398
G2J	2.2950	0.2417	8.0966	1.2586
G3J	2.1462	0.2597	8.3781	1.3023
G3H	2.3755	0.2584	8.5776	1.3333
G1H	2.2727	0.3543	10.7763	1.6751

The table above shows that the algorithms have not had decisive influence on the results with the exception of G2H; the method is the key element to be considered when measuring the performance of the procedures. Surprisingly, method K has turned out to be better than the rest for all methods. Also surprising are the relative performances. Relative to G3K (the best overall) the ratios for $G\alpha K$ are 1:1.02:1.05; the ones of $G\alpha P$ are 1.16:1.19:1.20; the one of $G\alpha S$ is 1.20 (recall that method S was only tried with G3); the ones of $G\alpha J$ are 1.24:1.26:1.30; and those of $G\alpha H$ are 1.17:1.33:1.67: with the exception of $G\alpha H$ all methods are closely clustered and seem to be insensitive to the algorithm. Also noteworthy is the fact that the relative difference between the best and worst performances is 67%.

5 Conclusions

From the previous results it follows that method K has yielded the best of all those we analyzed. This seems to contradict the reported experience regarding the mentioned methods. For example, in [8] the following is concluded:

"Penalties which are functions of the distance from feasibility are better performers than those which are merely functions of the number of violated constraints..."

Likewise, in [1], [9], [10] and [11] it is assumed that, as seems intuitively satisfying, those methods which take advantage of greater information are the most adequate to establish penalty functions. Interestingly, in none of these the validity of this intuitive dictum is proven either theoretically and/or experimentally. In [1], in fact, the conclusions (for only three functions) are contradictory; something which, in light of our results, is to be expected. Hence it is valid to ask whether this expectation is not satisfiable in general and, in fact, the results we found seem to experimentally refute the previously mentioned intuition. However, assuming that the intuitive assertion were correct, some possible explanations for the apparent anomaly are the following.

a) In this comparative analysis the same parameters for the penalty method were applied to all functions. Would it be more adequate, perhaps, to adjust the parameters particularly for every function?

b) It is possible that the combination of full elitism and the algorithm implemented in $G\alpha K$ maintain a varied population and this favors the efficiency of the method and that this accelerates, in general, the identification of the best value.

c) It is also possible that method K (in those cases where none of the method reaches the feasible region) does find points which satisfy a larger number of constraints in 100 generations while other methods are "closer" to satisfying all constraints.

This last hypothesis is illustrated in figure 1.

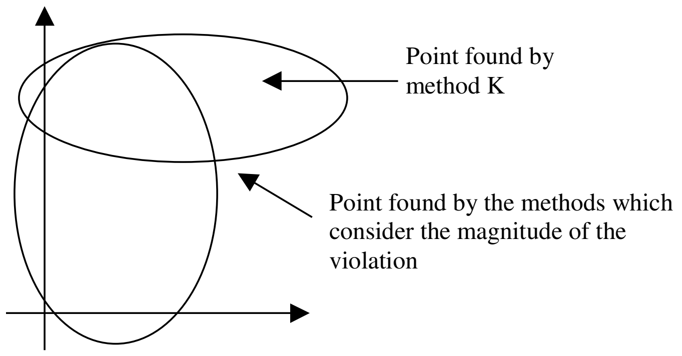
Point found by method K
Point found by the methods which consider the magnitude of the violation

Fig. 1. The point found by the methods which consider the size of the violation does not satisfy any constraint, whereas method K complies with one of them

If this were the situation we could conclude that method K finds the feasible points "trying" to satisfy the constraints one at a time and the methods which consider the magnitude of the constraint (as in three of the remaining methods) "try" to find the solution considering all constraints simultaneously.

d) It is possible that these results hold only for this suite of functions and that the suite displays this atypical behavior coincidentally. In this regard we must mention that we tested the calculated bounds with functions outside the suite and, as expected, the values are consistent with our study.

e) Finally, we may consider that method K is better because the suite consists (or is considered to consist) of "simple" functions. This argument depends on what we mean by a "simple" function. Although some of the functions in the suite would generally be considered to be in this category, some others certainly do not. At any rate, we may point out that if method K is better in those not-so-especially-difficult functions (those which we find on a day-to-day basis) then one possible conclusion is that it should be applied by default, resorting to some of the other methods only in case of a specifically complex or anomalous function.

There is much work still to be done. We plan to extend the study to a set of functions generated automatically in such a way that the possible bias in the suite is eliminated. In a mechanized set we also expect to make the difficulty inherent to the functions in the set more homogeneous.

References

1. Coello, C., “Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art", Computer Methods in Applied Mechanics and Engineering, 2001 (to be published).
2. Homaiffar A., Qi C. & Lai S., "Constrained Optimization Via Genetic Algorithms". Simulation, 62:4, pp. 242-254, 1994.
3. Joines, J and Houck, C., "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's". Proceedings of first IEEE Conference on Evolutionary Computation, pp. 579-584, 1994.
4. Shoenauer, M. and Xanthakis, S., "Constrained GA Optimization". Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 573-580, 1993.
5. Kuri, A., *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning. Theory and Applications*, Vol. 1. Instituto Politécnico Nacional, pp 270, 1999.
6. Powell, D. and Skolnick, M., "Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints". Proceedings of the Fifth International Conference on Genetic Algorithms. pp. 424-430, 1993.
7. Kuri, A., "A universal Eclectic Genetic Algorithm for Constrained Optimization". Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98, pp. 518-522, 1998.
8. Richardson J., Palmer M., Liepins G. & Hilliard M., “Some Guidelines for Genetic Algorithms with Penalty Functions”. Proceedings of the IEEE International Conference on Evolutionary Computation, pp.191-197, 1989.
9. Back, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.
10. Coello, C., “Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems”, Computers in Industry, 41(2):113-127, 2000.
11. Fogel, D., *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*, the Institute of Electrical and Electronic Engineers, New York, 1999.