SOME PENALTY METHODS WITH GENETIC ALGORITHMS

Janne Mäkinen, Kaisa Miettinen and Marko M. Mäkelä

University of Jyväskylä, Department of Mathematical Information Technology, P.O. Box 35 (MaE), FIN-40351 Jyväskylä, FINLAND

Abstract

We describe two constraint-handling methods to be used with genetic algorithms in global optimization. Both the methods, the method of superiority of feasible points and the method of adaptive penalties, are based on penalty function techniques. In addition, we introduce two modifications of these methods. We compare all the four methods numerically in nine test problems and report the results obtained.

1 INTRODUCTION

Genetic algorithms are well-known stochastic methods of global optimization based on the evolution theory of Darwin (see, for example, [Goldberg, 1989]). They have successfully been applied in different real-world applications, for example, in aeronautics, electrical engineering, scheduling and signal processing (see [Miettinen et al., 1999] among others).

In their basic form, genetic algorithms are not capable of handling constraint functions limiting the set of feasible solutions. Therefore, some additional methods are needed to keep the solutions in the feasible region. For a classification of these methods into seven different categories, see [Kim, Myung, 1997] and references therein.

Both direct and indirect methods have been proposed for handling constraints together with genetic algorithms. Usually, direct methods are applicable to less general problem types than indirect methods. Direct methods are mainly problem-dependent or they are restricted to certain types of functions.

As a rule, indirect constraint-handling methods are based on penalty function techniques where a penalty term is added to the objective function penalizing the function values outside the feasible region. This type of methods are popular because they are easy to implement beyond the underlying optimizer (like genetic algorithms). For a collection of both direct and indirect methods see, for example, [Michalewicz, Schoenauer, 1996] and references therein.

In this paper, we numerically compare two penalty function-based methods, the method of superiority of feasible points (see [Powell, Skolnick, 1993] and [Michalewicz, Schoenauer, 1996]) and the method of adaptive penalties (see [Hadj-Alouane, Bean, 1997] and [Michalewicz, Schoenauer, 1996]) in the context of genetic algorithms. In addition, we introduce two modifications to them. All these methods offer different approaches to constraint-handling. The comparison is realized by solving nine test problems from the literature.

2 ELEMENTS OF GENETIC ALGORITHMS

We consider an optimization problem of the form

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in S, \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function* and $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is a vector of *variables*. The *feasible region* S is bounded by box constraints and inequality constraints, that is,

$$S = \{x \in \mathbb{R}^n \mid x^l \leq x \leq x^u \text{ and } g_j(x) \leq 0 \text{ for } j = 1, \ldots, m\}, \tag{2.2}$$

where $x^l, x^u \in \mathbb{R}^n$ and $g_j : \mathbb{R}^n \to \mathbb{R}$ for $j = 1, \ldots, m$.

We are looking for the *global minimum* of the problem (2.1), in other words, the point $x^* \in S$ such that

$$f(x^*) \leq f(x) \text{ for all } x \in S.$$

An optimization problem is convex if both the objective function and the feasible region are convex (for definitions, see for example, [Bazaraa et al, 1993]). In convex problems, every optimal solution is a global optimum. This is not necessarily the case with more general, nonconvex, problems. Special methods of global optimization are needed in order to find global optima in the nonconvex case.

The basic idea behind genetic algorithms is to mathematically imitate the evolution process of nature. The algorithms are based on the evaluation of a set of solutions, called *population*. The population is treated with genetic operations.

At the iteration i the population X^i consists of a number of N *individuals* x^j, that is, solutions, where N is called a *population size*. The population is initialized by randomly generated individuals.

The individuals can be encoded using either binary or real numbers. We use the latter because of their popularity.

Each individual $x^j = (x^j_1, \ldots, x^j_n)^T$ is a vector of variables. Each variable is a real number. The suitability of an individual is determined by the value of the objective function, to be called a *fitness function*.

A new population is generated by the genetic operations *selection*, *crossover* and *mutation*. Parents are chosen by selection and new offsprings are produced with crossover and mutation. All these operations include randomness. The success of the optimization process is improved by *elitism* where the best individuals of the old population are copied as such to the next population. All the above-mentioned operations can be realized in many different ways see, for example, [Goldberg, 1989]. We employ so-called tournament selection and single-point crossover. The mutation is implemented in the way presented in [Mäkinen et al., to appear]. Note that the implementation of the genetic operations makes sure that box constraints are satisfied.

Genetic algorithm

The basic genetic algorithm can be presented in the form

1. Set the parameters of the genetic algorithm and the number of iterations to be carried out.
2. Initialize the population with random numbers.
3. Perform the operations selection, crossover, mutation and elitism in order to create a new population.
4. If the number of iterations has not been reached, return to step (3). Otherwise, choose the best individual found as the final solution.

3 HANDLING CONSTRAINTS

In their basic form, genetic algorithms can only handle box constraints. Naturally, it is always possible to reject those individuals that violate the general constraints, see for example, [Smith, 1984]. However, in particular, for large-scale problems this may be a very ineffective way.

Constraint functions can be taken into consideration by using the idea of penalty functions. This means that the constraints are included in the objective function in one way or the other. Leaving the feasible region is penalized by increasing the fitness function value. The increment depends on the distance to the feasible region.

3.1 METHOD OF SUPERIORITY OF FEASIBLE POINTS

The idea of the *method of superiority of feasible points* (SFP method) is presented in [Powell, Skolnick, 1993]. An example of the realization of the idea is given in [Michalewicz, Schoenauer, 1996]. In the SFP method, an additional function is added to the penalized objective function. This iteration-dependent function assures that infeasible solutions have higher fitness function values than feasible solutions.

The problem to be solved with a new fitness function is of the form

$$\begin{aligned} &\text{minimize} \quad \hat{f}(x) = f(x) + r\Big(\sum_{j=1}^{m} \max\big[0, g_j(x)\big]\Big) + \theta_k(x) \qquad (3.1) \\ &\text{subject to} \quad x^l \leq x \leq x^u, \end{aligned}$$

where

$$\theta_k(x) = \begin{cases} 0 & \text{if } X^i \cap S = \emptyset \ \text{ or } \ x \in S \\ \alpha & \text{otherwise} \end{cases} \qquad (3.2)$$

and

$$\alpha = \max\Big[0, \max_{y \in X^i \cap S} f(y) - \min_{z \in X^i \setminus S} \Big[f(z) + r\Big(\sum_{j=1}^{m} \max\big[0, g_j(z)\big]\Big)\Big]\Big]. \tag{3.3}$$

In other words, the function θ_k in (3.1) modifies the individuals of the population such that they cannot have better fitness function values outside the feasible region than what they have inside. The penalty coefficient r is here constant because the function θ_k discriminates between feasible and infeasible individuals.

Using the SFP method is a straightforward modification of the basic genetic algorithm. The main difference is the fitness function used. The stopping criterion is the number of iterations, that is, the number of populations to be generated i_{max}.

SFP algorithm

1. Set r, i_{max} and the population size N. Set the other genetic parameters. Set $i = 1$.

2. Generate N individuals to form the initial population X^1. Set $f_{best} = \infty$.

3. If the best individual in X^i is feasible and it gives the best fitness function value so far, update f_{best} and save that individual to x_{best}. If $i = i_{max}$, stop.

4. Set $X^{i+1} = \emptyset$. Carry out elitism. Repeat crossover and mutation until X^{i+1} has N individuals.

5. Set $i = i + 1$. Goto step 3.

If $f_{best} < \infty$, the final solution is x_{best}. Otherwise, the algorithm could not find any feasible solution.

3.2 METHOD OF ADAPTIVE PENALTIES

The *method of adaptive penalties* (AP method) tries to avoid infeasible solutions by adjusting the penalty coefficient. The method was originally proposed for multiple-choice integer programs in a report, later published in [Hadj-Alouane, Bean, 1997]. Anyhow, it can be generalized for other problems as, for example, in [Michalewicz, Schoenauer, 1996].

A new parameter h is used in the AP method. This parameter is the number of iterations whose best individuals are examined. If all the best individuals of the past h iterations are feasible, the penalty coefficient is decreased by dividing it with a parameter $c_1 > 1$. Otherwise, if all the best individuals of the past h iterations are infeasible, the penalty coefficient is increased by multiplying it with a parameter $c_2 > 1$. If some of the best individuals are feasible and some infeasible, we can continue with the current penalty coefficient. Thus, the penalty coefficient is updated if there is a possibility that the boundary of the feasible region is not covered or the search concentrates on infeasible solutions.

The problem to be solved with a new fitness function $\hat{f}$ is of the form

$$\begin{aligned} &\text{minimize} \quad \hat{f}(x) = f(x) + r_i\Big(\sum_{j=1}^{m} \max\big[0, g_j(x)\big]^2\Big) \\ &\text{subject to} \quad x^l \le x \le x^u, \end{aligned} \tag{3.4}$$

where the value of the penalty coefficient r_i is checked at each iteration i after the first h iterations. Let us denote the best individual of the iteration j by y^j. The coefficient is updated according to the formula

$$r_{i+1} = \begin{cases} \frac{r_i}{c_1} & \text{if } y^j \in S \text{ for all } i-h+1 \le j \le i \\ c_2 r_i & \text{if } y^j \notin S \text{ for all } i-h+1 \le j \le i \\ r_i & \text{otherwise,} \end{cases} \tag{3.5}$$

where $c_1, c_2 > 1$ and $c_1 \neq c_2$ (in order to avoid cycling).

AP algorithm

1. Set $c_1, c_2 > 1$ and r_1. Set h, i_{max}, population size N and the other genetic parameters. Set $i = 1$.
2. Generate N numbers to form the initial population X^1. Set $f_{best} = \infty$.
3. Save the best individual of X^i as y^i. If this individual is feasible and it gives the best fitness function value so far, update f_{best} and save that individual to x_{best}. If $i = i_{max}$, stop.
4. Set $X^{i+1} = \emptyset$. Carry out elitism. Repeat crossover and mutation until X^{i+1} has N individuals.
5. If $i < h$, set $r_{i+1} = r_i$ and goto step 3. If $i \ge h$ calculate r_{i+1} according to (3.5). Set $i = i + 1$ and goto step 3.

As in the SFP method, the final solution is x_{best} unless $f_{best} = \infty$. In the latter case, the algorithm did not manage to find any feasible solution.

The magnitude of the constants c_1 and c_2 is examined in [Hadj-Alouane, Bean, 1997]. It is shown that the method is rather robust with respect to these values. Some kind of a recommendation is to set $c_1 = 4$ and $c_2 = 2.8$.

Some formulas are given in [Hadj-Alouane, Bean, 1997] also for the determination of h and r_1. However, they are specific to the multiple-choice integer program.

3.3 MODIFICATIONS

The idea of the AP method is to update the penalty coefficient based on the feasibility of the best individuals of h iterations. In this approach, the feasibility of the whole population is not considered. On the other hand, if the best solution of a population is infeasible whereas the second best solution is feasible, the best (feasible) solution found is not updated. This is the case even if the second best solution would be better than the current x_{best}. This feature is a theoretical drawback of the method.

For these reasons, the method was modified so that the best feasible solution found is recorded and the penalty coefficient is updated if the whole population lies inside or outside the feasible region. The update is not necessary to be carried out at every iteration. That is why we need a parameter for the frequency how often the feasibility is checked. Note that the parameter h is not needed in this approach. In what follows, this modification of the AP method is called MAP. In all the other respects, the MAP algorithm corresponds to the AP algorithm.

It may happen that the SFP method is unable to find any feasible solutions. If the initial population does not contain any feasible solutions, the constraint-handling part of the fitness function does not direct the population towards the feasible region because the penalty coefficient is kept constant. That is why it is interesting to combine the modified adaptive penalty coefficient (MAP) technique and the SFP method. This combination will be called MASF. Note that a combination of the original AP and the SFP methods is not sensible because the update process checks only the best individuals and they are always feasible in the SFP method (assuming they exist).

4 NUMERICAL EXPERIMENTS

Numerical experiments were carried out to test the performance of the constraint-handling methods described. Nine test problems of different types were selected from the literature. Because of the stochastic nature of genetic algorithms, the calculations were performed a hundred times for each problem.

Problem 1 was set to be

$$\begin{aligned} &\text{minimize} && -\left(x + x^2 - \frac{1}{x^2} + x^3\right) \\ &\text{subject to} && 100x \leq 0 \\ & && -10.0 \leq x \leq 10.0 \end{aligned}$$

whereas problem 2 was selected as

$$\begin{aligned} &\text{minimize} && 2\sin(x) + \sqrt{x} \\ &\text{subject to} && 3 - 0.3x^2 \leq 0 \\ & && 0.0001x^4 - 10 \leq 0 \\ & && 0.0 \leq x \leq 20.0. \end{aligned}$$

As far as problems 3, 4 and 5 are concerned, see problems 1, 4 and 5 in [Kim, Myung, 1997], respectively. Problems 6–9 can be found in [Michalewicz, Schoenauer, 1996] as G9, G1, G6 and G7, respectively.

All the methods were implemented in Fortran77 and the test runs were performed on an HP9000/C160, 160MHz computer.

Because the idea was to compare different constraint-handling techniques, the input parameters of the genetic algorithm were constant in all the calculations. The population size N was 100 and the number of populations i_{max} was 250.

In the SFP method, the penalty coefficient r was 10.0 and in all the other methods the initial penalty coefficient r_1 was 1.0. In the AP method, the number of iterations whose best individuals are examined h was 10. The constants in the adaptive penalty methods were selected as $c_1 = 4.0$ and $c_2 = 2.8$.

As far as the modified methods are concerned, the penalty coefficient was updated every 25 iterations.

Each of the nine test problems was solved for one hundred times. We recorded the best objective function values for each test run. In what follows, we list the means of these values. For the convenience of the reader, we also write down the best known objective function values.

Table 1 contains the mean values of the results obtained with the SFP, AP, MAP and MASF methods.

Prob.	**Optimum**	**SFP**	**AP**	**MAP**	**MASF**
1	2.000	2.000	2.000	2.000	2.000
2	0.157	0.157	0.157	0.157	0.157
3	0.250	0.349	0.349	0.288	0.275
4	5.000	5.509	5.169	5.132	5.284
5	1.000	1.000	1.000	1.050	1.000
6	680.630	685.219	685.399	684.304	684.975
7	-15.000	-13.534	-13.975	-14.452	-14.472
8	-6961.814	-4125.447	-5436.201	-5849.261	-5332.740
9	24.306	33.547	30.884	29.282	32.726

Table 1. Results with four different methods.

With the SFP method, a feasible solution could only be found in 6% of the tests in problem 8. The corresponding number for the AP method was 56%, for the MAP method 92% and for the MASF method 81%.

5 DISCUSSION AND CONCLUSIONS

The methods are not easy to compare because of their stochastic nature. We have decided to compare the average performances. No strict superiority can be expressed. From Table 1 we can deduce that the modified methods MAP and MASF managed to find somewhat better solutions than the AP and SFP methods. In particular, the MAP method proved out to be best in the last, most difficult, problems.

In addition, the modified methods were able to find feasible solutions to problem 8 more often than the other methods. The worst method for this problem was the SFP method, which managed to find feasible solutions in only 6 cases of 100.

The differences in the CPU times used are not significant partly because of the relative simplicity of the test problems and partly because of the selected stopping criterion. In other words, a number of 250 populations were always generated no matter how the search process proceeded. In general, AP and MAP were slightly faster than SFP and MASF.

Here, the way of implementing genetic operations was not compared even though it may explain, for example, the difficulties of the SFP method in finding feasible solutions. This is a subject of future research.

REFERENCES

1. M.S. Bazaraa, H.D. Sherali, C.M. Shetty. Nonlinear Programming: Theory and Algorithms. John Wiley & Sons, 2nd edition, 1993.

2. D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, 1989.

3. A.B. Hadj-Alouane, J.C. Bean. A Genetic Algorithm for the Multiple-Choice Integer Program, *Operations Research*, 45, no. 1 (1997), pp. 92–101.

4. J.-H. Kim, H. Myung. Evolutionary Programming Techniques for Constrained Optimization Problems, *IEEE Transactions on Evolutionary Computation*, 1, no. 2 (1997), pp. 129–140.

5. R.A.E. Mäkinen, J. Toivanen, J. Périaux. Multidisciplinary Shape Optimization in Aerodynamics and Electromagnetics Using Genetic Algorithms, *International Journal for Numerical Methods in Fluids*, to appear.

6. Z. Michalewich, M. Schonauer. Evolutionary Algorithms for Constraint Parameter Optimization Problems, *Evolutionary Computation*, 4, no. 1 (1996), pp. 1–32.

7. K. Miettinen, M.M. Mäkelä, P. Neittaanmäki, J. Périaux (eds). Evolutionary Algorithms in Engineering and Computer Science, John Wiley & Sons, 1999.

8. D. Powell, M.M. Skolnick. Using Genetic Algorithms in Engineering Design Optimization with Non-Linear Constraints. Proceedings of the Fifth International Conference on Genetic Algorithms, edited by S. Forrest, Morgan Kaufmann Publishers, 1993, pp. 424–431.

9. R.L. Smith. Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed over Bounded Regions, *Operations Research*, 32, no. 6 (1984), pp. 1296–1308.