On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's

Jeffrey A. Joines and Christopher R. Houck
jjoine@eos.ncsu.edu chouck@eos.ncsu.edu
Department of Industrial Engineering
North Carolina State University, NC 27695-7906

Abstract

In this paper we discuss the use of non-stationary penalty functions to solve general nonlinear programming problems (NP) using real-valued GAs. The non-stationary penalty is a function of the generation number; as the number of generations increases so does the penalty. Therefore, as the penalty increases it puts more and more selective pressure on the GA to find a feasible solution. The ideas presented in this paper come from two basic areas: calculus-based nonlinear programming and simulated annealing. The non-stationary penalty methods are tested on four NP test cases and the effectiveness of these methods are reported..

1 Introduction

Constrained function optimization is an extremely important tool used in almost every facet of engineering, operations research, mathematics, and etc. Constrained optimization can be represented as a nonlinear programming problem. The general nonlinear programming problem is defined as follows:

$$(NP) \qquad \textit{minimize} \qquad f(\overline{X})$$

subject to (nonlinear and linear) constraints:

$$\begin{aligned} g_i(\overline{X}) &\geq 0.0, \quad i = 1, \ldots, m \\ h_j(\overline{X}) &= 0.0, \quad j = 1, \ldots, p \end{aligned}$$

In the last few years, there has been a growing effort to apply genetic algorithms (GA) to general constrained optimization problems [5, 6, 7]. GAs have been widely applied to unconstrained optimization where their appeal is their ability to solve ill-conditioned problems. Traditional calculus-based or deterministic global search methods typically make strong assumptions regarding the objective function, i.e., continuity, differentiability, satisfaction of the Lipschitz Condition, etc., in order to make the search method justifiable. These conditions also hold for any linear and nonlinear constraints of a constrained optimization problem. It is our expectation that GAs can overcome these limitations as well.

This paper is organized into four sections. The following section, Section 2, describes existing methods used to solve constraint optimization problems with GAs. Section 3 introduces and explains the concepts of non-stationary penalty functions used in solving general nonlinear programming problems. Section 4 details the results obtained from applying these techniques to four test cases. The last section of this paper will state the conclusions developed from the experiments and the direction of future research.

2 Previous GA Constrained Optimization Methods

Traditional GAs have encompassed constraints in the form of bounds on the variables. However, only recently have researchers begun to attack general constrained optimization problems. The difficulty of using GAs in constraint optimization is that the genetic operators used to manipulate the chromosomes of the population often produce solutions which are not feasible. Presently, there are four methods used to handle constraints with GAs: rejection of the offspring, repair algorithms, modified genetic operators, and penalty functions.

When infeasible offspring have been created, these offspring can be rejected from entering the population. This technique can spend a great deal of time in the evaluation and rejection of these infeasible solutions. When an infeasible solution has been created by an operator, special repair algorithms for that operator can

0-7803-1899-4/94 $4.00

be employed to restore feasibility. However, repair algorithms are problem specific, the children often do not resemble their parents, and restoring feasibility may be as difficult as the optimization problem [5].

Michalewicz has developed a system, GENOCOP, that can solve any linear constrained problem using a set of special genetic operators.The operators designed in the system are designed to exploit the convex region produced by the linear constraints. Any linear combination of two feasible points in a convex region will produce another feasible point [5, 6]. The limitation of this system is that it cannot handle nonlinear constraints because they do not necessarily produce convex regions.

Penalty function techniques transform the constrained problem into an unconstrained problem by penalizing those solutions which are infeasible. It has been shown that penalty functions based on the distance from feasibility outperform those based upon the number of violated constraints [7]. The main limitation of the penalty functions is the degree to which each constraint is penalized. Researchers have stated that if one imposes a high degree of penalty, more emphasis is placed on obtaining feasibility and the GA will move very quickly towards a feasible solution. The system will tend to converge to feasible point even if it is far from optimal. However, if one impose a low degree of penalty, less emphasis is placed on feasibility, and the system may never converge to a feasible solution [2, 5].

3 Non-Stationary(NS) Penalty Methods

The use of penalty methods to solve nonlinear programming problems originated in 1943 by Courant. Courant, however, used these penalty methods only to obtain solutions to differential equations. It was Fiacco and McCormick, and Zangwill who started applying these penalty methods to general NP problems in the late 1960's [1, 2]. These penalty methods solve the general NP problem through a sequence of unconstrained optimization problems. These methods force infeasible points toward the feasible region by step-wise increasing the penalty, ρ_k used in the penalizing function, $\mathcal{P}(\rho_k, \overline{X})$. It has been shown that a solution, $\overline{X}$ which minimizes NP' also minimizes NP as, k approaches infinity:

$$(NP') \qquad minimize \qquad F(\overline{X}, \rho_k) = f(\overline{X}) + \mathcal{P}(\rho_k, \overline{X})$$

where:

$$\lim_{k \to \infty} \quad \rho_k = \infty \qquad \lim_{\overline{X} \to \text{feasibility}} \quad \mathcal{P}(\rho_k, \overline{X}) = 0$$

3.1 Non-Stationary Methods with GA's

This sequential unconstrained optimization approach could be used for genetic algorithms. Michalewicz has used such an approach in his GENOCOP II system [6]. This system runs a genetic algorithm with a constant penalty, then increases the penalty and runs the GA again. This procedure repeats until an acceptable solution is found. The GENOCOP II system produces a series of solutions by initially optimizing the unconstrained problem and then gradually optimizing the constrained function. We incorporate a step-wise penalty increases by increasing the penalty, ρ_k, within a run of the genetic algorithm.

$$\rho_k = C \times k, \quad C = \text{Constant and } k = \text{Generation } \#$$

Our penalizing function, $\mathcal{P}(\rho_k, \overline{X})$, is based on the sum of the violated constraints, $SVC(\beta, \overline{X})$.

$$D_i(\overline{X}) = \begin{cases} 0 & g_i(\overline{X}) \geq -\epsilon \\ |g_i(\overline{X})| & \text{otherwise} \end{cases} \qquad 1 \leq i \leq m$$

$$D_j(\overline{X}) = \begin{cases} 0 & -\epsilon \leq h_j(\overline{X}) \leq \epsilon \\ |h_j(\overline{X})| & \text{otherwise} \end{cases} \qquad 1 \leq j \leq p$$

$$SVC(\beta, \overline{X}) = \sum_{i=1}^{m} D_i^{\beta}(\overline{X}) + \sum_{j=1}^{p} D_j^{\beta}(\overline{X}), \qquad \beta = 1, 2, \ldots$$

We are introducing a family of penalty functions based up traditional calculus-based methods.

$$\mathcal{P}(\alpha, \beta) = \rho_k^{\alpha} \times SVC(\beta, \overline{X})$$

Additionally, we tried a family of methods where their origin comes from simulated annealing.

$$\mathcal{R}(\alpha,\beta) = e^{\mathcal{P}(\alpha,\beta)}$$

Four variations of the penalty methods were tested. The first three penalty methods were based on the traditional family: $\mathcal{P}(1,1)$, $\mathcal{P}(1,2)$, and $\mathcal{P}(1,2)$. The fourth variation was based on the simulated annealing family, $\mathcal{R}(1,1)$.

4 Test Cases

To test the performance of the system, four test cases were selected and were evaluated using several criterion. The average and standard deviation of the best solution from each of the 10 replications is reported as well as the average of the sum of the violated constraints of the best solution for each run. The %Feasible is the percent of the runs in which the best solution was a feasible solution. The "Best" (Distance) is the minimum solution of the 10 runs and its corresponding sum of constraint violations. The "Best Feasible"(Distance) is the most feasible solution from the 10 runs and its corresponding sum of constraint violations.

A floating point representation was used along with geometric ranking selection with normalization. The same six operators were used as were in GENOCOP [5, 6]. However, for non-linear constraints, they do not maintain feasibility. For each test case, we made 10 replications using the same random seed for each method. All runs were performed with the following GA parameters: $popsize = 80$, $k = 28$ (number of parents in each generation), $b = 2$ (coefficient for non-uniform mutation), $a = 0.25$ (parameter of arithmetical crossover), $C = 0.5$ and 0.05 (*constant for* $\mathcal{P}$ *family and* $\mathcal{R}$ *family respectively*).

We have been able to solve several problems similar to that of Test Case #1 involving two to three variables and constraints. Therefore, the set of problems was chosen to illustrate the potential of the proposed methods to solving difficult problems. These problems include nonlinear equalities, involving up to 8 variables and 6 binding constraints.

4.1 Test Case #1

This problem (taken from [3]) is

$$\textit{minimize} \quad f(\overline{X}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to the nonlinear constraints:

$$\begin{aligned} (x_1 - 5)^2 + (x_2 - 5)^2 - 100 &\geq 0.0 \\ -(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 &\geq 0.0 \end{aligned}$$

and bounds: $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$.

The known global solution is $\overline{X}^* = (14.095, 0.84296)$ and $f(\overline{X}^*) = -6961.81381$. $\mathcal{P}(1,1)$,$\mathcal{P}(2,2)$, and $\mathcal{R}(1,1)$ consistently returned the optimal answer. However, $\mathcal{P}(1,2)$ had problems due to the selective pressure being too small.

	$\mathcal{P}(1,1)$	$\mathcal{P}(1,2)$	$\mathcal{P}(2,2)$	$\mathcal{R}(1,1)$
Avg.	-6960.0418	-7017.6853	-6958.3841	-6960.985
Std. Dev.	0.96693163	4.9265354	0.96693163	0.99235469
Avg Dist.	0.0	0.0481	0.0	1.68×10^{-6}
% Feasible	100%	0%	100%	90%
Best	-6961.7853	-7028.0926	-6961.2399	-6961.8253
(Distance)	-(0.0)	(0.05711)	(0.0)	(1.68×10^{-5})
Best Fease.	-6961.7853	-7011.2646	-6961.2399	-6961.7853
(Distance)	(0.0)	(0.04252)	(0.0)	(0.0)

Table 1: Results for Test Case #1

4.2 Test Case #2

This problem (problem 100 taken from [4]) is

$$\begin{aligned} \textit{minimize} \quad f(\overline{X}) &= (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+ \\ &\quad 10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7 \end{aligned}$$

subject to the nonlinear constraints:

$$\begin{aligned} 127-2x_1^2-3x_2^4-x_3-4x_4^2-5x_5 &\geq 0.0 \\ 282-7x_1-3x_2-10x_3^2-x_4+x_5 &\geq 0.0 \\ 196-23x_1-x_2^2+-6x_6^2+8x_7 &\geq 0.0 \\ -4x_1^2-x_2^2+3x_1x_2-2x_3^2-5x_6+11x_7 &\geq 0.0 \end{aligned}$$

The best known solution to the problem has a value of $f(\overline{X}^*) = 680.6400573$ which is infeasible by the sum of violated constraints was 0.90×10^{-7} which is less than ϵ, the criteria used for machine precision.

	$\mathcal{P}(1,1)$	$\mathcal{P}(1,2)$	$\mathcal{P}(2,2)$	$\mathcal{R}(1,1)$
Avg.	683.6809	684.4679	684.9892	680.91724
Std. Dev.	1.5237456	2.1024116	2.1024116	0.41798344
Avg. Dist.	0.0	0.0	0.0	0.1322
% Feasible	100%	100%	100%	0%
Best	681.76981	682.23860	681.45717	680.4336
(Distance)	(0.0)	(0.0)	(0.0)	(0.2178)
Best Fease.	681.76981	682.23860	681.45717	680.8643
(Distance)	(0.0)	(0.0)	(0.0)	(0.04986)

Table 3: Results for Test Case #3

$\mathcal{P}(1,1)$, $\mathcal{P}(1,2)$, and $\mathcal{P}(2,2)$found solutions which were completely feasible and only 0.12% higher than the known best solution. The $\mathcal{R}(1,1)$, penalizing function encountered difficulties due to the initial sum of the violated constraints being very large, thus leading to overflows.

4.3 Test Case #3

This problem (problem 100 taken from [4]) is

$$\textit{minimize} \quad f(\overline{X}) = 3x_1+1.0\text{x}10^{-6}x_1^3+2x_2+\frac{2}{3}\text{x}10^{-6}x_2^3$$

subject to the linear and nonlinear constraints:

$$\begin{aligned} x_4-x_3+0.55 &\geq 0.0 \\ x_3-x_4+0.55 &\geq 0.0 \end{aligned}$$

$$\begin{aligned} 1000sin(-x_3-0.25)+1000sin(-x_4-0.25)+894.8-x_1 &= 0.0 \\ 1000sin(x_3-0.25)+1000sin(x_3-x_4-0.25)+894.8-x_2 &= 0.0 \\ 1000sin(x_4-0.25)+1000sin(x_4-x_3-0.25)+1294.8 &= 0.0 \end{aligned}$$

and bounds: $0 \leq x_i \leq 1200$, $i = 1,2$ and $-0.55 \leq x_i \leq 0.55$, $i = 3,4$

The best known optimal for Test Case #4 is $f(\overline{X}^*) = 5126.4981$. According to [8] this point was not completely feasible because the sum of the violated constraints was 0.75×10^{-7} which was still less than ϵ.

	$\mathcal{P}(1,1)$	$\mathcal{P}(1,2)$	$\mathcal{P}(2,2)$	$\mathcal{R}(1,1)$
Avg.	5317.8204	5358.4130	5520.0903	5207.9604
Std. Dev.	199.60063	366.8887	364.7738	72.616
Avg. Dist..	0.004203	0.0008779	0.001434	0.005642
% Feasible	0%	0%	0%	0%
Best	5127.7135	5126.6653	5126.4920	5128.8100
(Distance)	(0.001188)	(0.001625)	(0.002569)	(0.02922)
Best Fease.	5137.0945	6056.8238	6108.1802;	5239.9019
(Distance)	(0.0002236)	(0.000189)	9.198×10^{-5}	(0.0005527)

Table 4: Results for Test Case #4

No method returned a feasible solution due to the three equality constraints. If we had manipulated the constants used, or allowed the simulation to run longer, we may have been able to achieve a more feasible point.

4.4 Test Case #4

This problem (problem 369 taken from [8]) is

$$\textit{minimize} \quad f(\overline{X}) = x_1 + x_2 + x_3$$

subject to the linear and nonlinear constraints:

$$\begin{aligned}
1 - 0.0025x_4 - 0.0025x_6 &\geq 0.0 \\
1 - 0.0025x_5 - 0.0025x_7 + 0.0025x_4 &\geq 0.0 \\
1 - 0.01x_8 + 0.01x_5 &\geq 0.0 \\
1 - \frac{833.33252x_4}{x_1x_6} - \frac{100}{x_6} + \frac{83333.333}{x_1x_6} &\geq 0.0 \\
1 - \frac{1250x_5}{x_2x_7} - \frac{x_4}{x_7} + \frac{1250x_4}{x_2x_7} &\geq 0.0 \\
1 - \frac{1250000}{x_3x_8} - \frac{x_5}{x_8} + \frac{2500x_5}{x_3x_8} &\geq 0.0
\end{aligned}$$

and bounds: $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$, $i = 2, 3$, and $10 \leq x_j \leq 1000$, $j = 4, 5, 6, 7, 8$.

The best known solution is 7049.24. According to [8] this point was not completely feasible. Again the sum of the violated constraints was 0.234×10^{-7} which was less than ϵ. This was the hardest test case due to the fact that all the constraints are binding.

	$\mathcal{P}(1,1)$	$\mathcal{P}(1,2)$	$\mathcal{P}(2,2)$	$\mathcal{R}(1,1)$
Avg.	7244.2786	4181.3247	7535.0518	7732.7845
Std. Dev.	107.75158	50.077184	257.78297	571.75755
Avg. Dist.	0.0	0.5079	0.0	0.0
% Feasible	100%	0%	100%	100%
Best	7068.6880	4112.324	7235.8714	7173.7845
(Distance)	(0.0)	(0.505)	(0.0)	(0.0)
Best Fease.	7068.6880	4245.2536	7235.8714	7173.7845
(Distance)	(0.0)	(0.4732)	(0.0)	(0.0)

Table 5: Results for Test Case #5

This problem is by far the most complex function analyzed. Even though the objective function is simple, all six constraints are binding. $\mathcal{P}(1,1)$,$\mathcal{P}(2,2)$, and $\mathcal{R}(1,1)$ all performed well, returning completely feasible solutions approximately 0.27% above the best known solution. $\mathcal{P}(1,2)$ had the same difficulties that it encountered in Test Case #1.

5 Conclusions

Penalty functions have been used to transform a constrained optimization problem into an unconstrained optimization problem. Traditional calculus-based penalty methods gradually increase the penalty to obtain the optimal feasible value. We have incorporated this concept of a non-stationary penalty function by increasing the penalty proportionally to the generation number. The goal is to allow the GA to explore more of the space before confining to it the feasible region. However, if enough pressure is not placed on the GA, it may never converge to a feasible solution. Therefore, the constant C plays an important role of controlling the convergence rate.

When comparing the two traditional methods, $\mathcal{P}(1,1)$ and $\mathcal{P}(1,2)$ the following observation was noted. As the constraints become barely violated, (i.e. distance from feasibility $\ll 1$), it takes a greater selective pressure to force that solution to feasibility. Test Case #1 and Test Case #4 demonstrated this fact, none of the points were feasible using just $\mathcal{P}(1,2)$. $\mathcal{P}(2,2)$ squares the generation number putting a greater selective pressure forcing $\mathcal{P}(1,2)$ to converge to feasible solutions. $\mathcal{P}(1,1)$ on average did better than the other methods. However, it takes longer to converge to a feasible optimal solution than the other three methods.

With all four methods, we encountered problems with constraint normalization. This was especially apparent in Test Case #3 with the $\mathcal{R}(1,1)$ method. Initial points in the population violated one constraint by 1×10^{10}, which would lead to overflow errors for the exponential. This also led to this constraint dominating the GA for all methods. Therefore, we would like to extend this method to include some constraint normalization techniques.

For each test case the parameters were fixed. There was no attempt to optimize any of the standard parameters associated with GAs, or the penalty methods. By simply running the GA longer, we are guaranteed to obtain a more feasible solution. We would like to examine the effect the parameters have on obtaining feasible optimal solutions (i.e. the constant C).

In GENOCOP II and GENOCOP, linear constraints are handled through special genetic operators. We would like to adopt this idea to see if it would help with solving problems with linear constraints like Test Case #3 and Test Case #4. As already stated, the linear constraints produce a convex hull and the genetic operators will force the GA to stay in this convex hull. This convex hull will act as limiting covering convex set of the nonlinear region. This confines the search space to this hull which should help with convergence and optimally. Other future directions include the use of double elitist model where not only the best individual is saved from generation to generation but also the best feasible point is kept.

References

[1] Avriel, M., Nonlinear Programming Analysis and Methods, Prentice-Hall, Inc., Englewood Cliffs, 1976.

[2] Bazaraa, M. S., Nonlinear programming : theory and algorithms, Wiley, New York, 1979.

[3] Floudas, C.A. and Pardalos, P.M., A Collectin of Test Problems for Constrained Global Opti mization Algorithms, Springer-Verlag, Lecture Notes in Computer Science, Vol. 455, 1987.

[4] Hock, W. and Schittkowski, K., Test Examples for Nonlinear Programming Codes, Springer-Verlag, Lecture Notes in Economics and Mathematical Systems, Vol. 187, 198 .

[5] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, AI Series, New York, 1992.

[6] Michalewicz, Z and Attia, N, Genetic Algorithm + Simulated Annealing = GENOCOP II: A Tool for Nonlinear Programming, submitted for publication.

[7] Richardson, J., Palmer, M., Liepins, G. and Hilliard, M., Some Guidelines for Genetic Algorithms with Penalty Functions, Proceedings of the Third International Conference on Genetic Algorithms, 1989.

[8] Schittkowski, K., More Test Examples for Nonlinear Programming Codes, Springer-Verlag, Lecture Notes in Economics and Mathematical Systems, Vol. 282, 198 .