The use of a multiobjective optimization technique to handle constraints

Carlos A. Coello Coello[a]

[a] *Laboratorio Nacional de Informática Avanzada*
Rébsamen 80, Xalapa, Veracruz 91090, México
E-mail: ccoello@xalapa.lania.mx

This paper presents a new approach to handle constraints using evolutionary algorithms. The new technique treats constraints as objectives, and uses a multiobjective optimization approach to solve the re-stated single-objective optimization problem. The new approach is compared against other numerical and evolutionary optimization techniques in several engineering optimization problems with different kinds of constraints. The results obtained show that the new approach can consistently outperform the other techniques without a significant sacrifice in terms of performance.

Keywords: genetic algorithms, constraint handling, evolutionary optimization.

1. Introduction

Despite the well-documented success of genetic algorithms (GAs) in a wide range of applications, their use in constrained optimization problem still raises several issues to which a considerable amount of research has been devoted in the last few years. From these, the key issue is how to incorporate constraints of any sort (linear, non-linear, equality or inequality) into the fitness function as to guide the search properly.

For several years, practitioners have used penalty functions to incorporate (mainly inequality) constraints into the fitness function, and there have been a lot of successful applications of this approach in all engineering fields. However, penalty functions have some well-known limitations [2], from which the most remarkable is the difficulty to define good penalty factors. These penalty factors are normally generated by trial and error, although their definition may severely affect the results produced by the GA [2].

In this paper, a new constraint-handling approach is proposed that does not require the use of a penalty function to handle equality and inequality constraints. This technique is based on a multiobjective optimization approach, and it is very suitable for parallelization.

The remainder of this paper is organized as follows: first, the new approach will be introduced, then two engineering optimization problems will be presented, and solved using the new approach. After that, the results produced by other (GA-based and mathematical programming) techniques will be compared with those obtained with the new method, and finally there will be some discussion of the results obtained and the expected paths of future research.

2. Description of the new approach

The main idea behind the approach proposed in this paper is to use a population-based multiobjective optimization technique such as VEGA [4] to handle each of the constraints as an objective in the way indicated before. The technique may be better illustrated by Figure 1. At each generation, the population is split into $m+1$ sub-populations, where m refers to the number of constraints of the problem (we have to add one to consider also the objective function). Although the size of each sub-population may be variable, it was

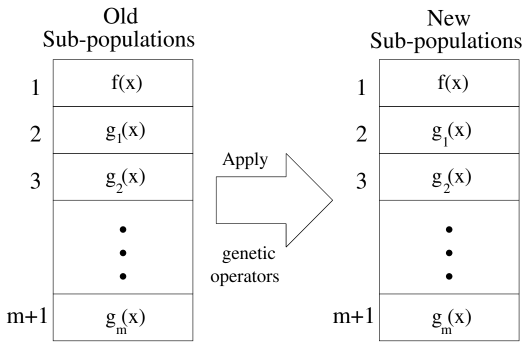

Figure 1. Graphical representation of the approach introduced in this paper.

decided to allocate the same size to each of them in the experiments reported in this paper, but the use of different sizes remains as an open issue that requires further research.

Using this scheme, a fraction of the population will be selected using the (unconstrained) objective function as its fitness; another fraction will use the first constraint as its fitness and so on. However, it is not completely obvious how to guide each of these sub-populations during the search.

For the sub-population guided by the objective function, the evaluation of such objective function for a given vector $\mathbf{X}$ (decoded from the chromosome) is used directly as the fitness function (probably multiplied by (-1) if it is a minimization problem), with no penalties of any sort. For all the other sub-populations, the algorithm used was the following:

$$\begin{array}{lll} \textbf{if } g_j(\mathbf{X}) < 0.0 & \textbf{then} & \text{fitness} = g_j(\mathbf{X}) \\ \textbf{else if } v \neq 0 & \textbf{then} & \text{fitness} = -v \\ \textbf{else} & & \text{fitness} = f \end{array}$$

where $g_j(\mathbf{X})$ refers to the constraint corresponding to sub-population $j+1$ (this is assuming that the first sub-population is assigned to the objective function f), and v refers to the number of constraints that are violated ($\leq m$). Finally, if the solution encoded is feasible, then this individual will be 'merged' with the first sub-population, since it will be evaluated with the same fitness function (i.e., the objective function).

There are a few interesting things that can be observed from this procedure. First, each sub-population associated with a constraint will try to reduce the amount in which that constraint is violated. If the solution evaluated does not violate the constraint corresponding to that sub-population, but it is infeasible, then the sub-population will try to minimize the total number of violations, joining then the other sub-populations in the effort of driving the GA to the feasible region. This aims at combining the distance from feasibility with information about the number of violated constraints, which is the same heuristic normally used with penalty functions. However, traditionally it is necessary to define in advance either an static penalty value or a dynamic penalty function that estimates this distance from feasibility, whereas in the current approach such distance is estimated automatically by the above algorithm using the constraint violation information derived from the GA run.

It is important to clarify that the current approach does not use dominance to impose an order on the constraints based on their violation (like in the case of COMOGA [3]) which is a more expensive process (in terms of CPU time) that also requires additional parameters. In fact, the current approach does not rank individuals, but it uses instead different fitness functions for each of the sub-population allocated (whose number depends on the number of constraints) depending on the feasibility of the individuals contained within each of them. This is easier to implement, does not require special operators to preserve feasiblity (like in the case of Parmee and Purchase's approach [5]), makes unnecessary the use of a sharing function to preserve diversity (like with traditional multiobjective optimization techniques), and does not require extra parameters to control the mixture of feasible and infeasible individuals (like in the case of COMOGA [3]).

It is interesting to notice that the use of the unconstrained objective function in one of the sub-populations may assign good fitness values to infeasible individuals. However, because the number of constraints will normally be greater than one, the other sub-populations will drive the GA to the feasible region. In fact, the sub-population evaluated with the objective function will be useful to keep diversity in the population, making then unnecessary the use of sharing techniques. The behavior expected under this scheme is to have few feasible individuals at the beginning, and then gradually produce solutions that may be feasible with respect to some constraints but not with respect to others. Over time, the building blocks of these sub-populations will combine to produce individuals that are feasible, but not necessarily optimum. At that point the direct use of the objective function will help the GA to approach the optimum, but since some infeasible solutions will still be present in the population, those individuals will be responsible to keep the diversity required to avoid stagnation.

The new approach has been tested with several functions of different degrees of difficulty, and has been able to deal properly with both inequality and equality constraints. However, due to space limitations, only 2 examples will be shown. The interested reader may consult [6] for further information.

3. Examples

3.1. *Example 1 : Design of a hydrostatic thrust bearing*

In this problem we want to minimize the power loss during the operation of a hydrostatic thrust bearing (see Figure 2) which has to withstand a specified load while providing an axial support. Four design variables are considered: R (bearing step radius), R_0 (recess radius), μ (viscosity), and Q (flow rate). The optimization problem can be stated as follows:

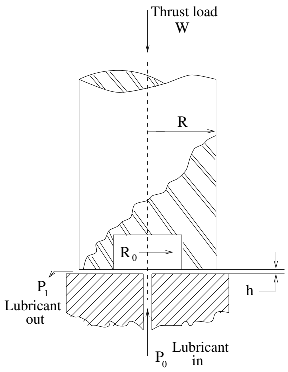

Figure 2. The hydrostatic thrust bearing used for the first example.

Minimize :

$$F(\mathbf{X}) = \frac{QP_0}{0.7} + E_f \tag{1}$$

Subject to :

$$g_1(\mathbf{X}) = \frac{\pi P_0}{2} \frac{R^2 - R_0^2}{\ln(R/R_0)} - W_s \geq 0 \tag{2}$$

$$g_2(\mathbf{X}) = P_{max} - P_0 \geq 0 \tag{3}$$

$$g_3(\mathbf{X}) = \Delta T_{max} - \Delta T \geq 0 \tag{4}$$

$$g_4(\mathbf{X}) = h - h_{min} \geq 0 \tag{5}$$

$$g_5(\mathbf{X}) = R - R_0 \geq 0 \tag{6}$$

$$g_6(\mathbf{X}) = 0.001 - \frac{\gamma}{gP_0}\left(\frac{Q}{2\pi Rh}\right) \geq 0 \tag{7}$$

$$g_7(\mathbf{X}) = 5000 - \frac{W}{\pi(R^2 - R_0^2)} \geq 0 \tag{8}$$

where the inlet pressure P_0 is defined as

$$P_0 = \frac{6\mu Q}{\pi h^3} \ln \frac{R}{R_0} \tag{9}$$

and the power loss due to friction is

$$E_f = 9336.0 Q \gamma C \Delta T \tag{10}$$

where $\gamma = 0.0307$ lb/in^3 (849.5755 kg/m^3) is the weight density of oil and $C = 0.5$ Btu/lb °F (0.5 cal/g °C) is the specific heat of oil. The temperature rise ΔT can be calculated from $\Delta T = 2P_2$, where

$$P_2 = 10^{P_3} - 560 \tag{11}$$

and

$$P_3 = \frac{\log_{10}\log_{10}(8.122 \times 10^6 \mu + 0.8) - C_1}{n} \tag{12}$$

C_1 and n are constants for a given oil. For the purposes of this example, we will use SAE 20 grade oil, with $C_1 = -3.55$ and $n = 10.04$. After calculating the value of E_f, we can calculate the film thickness h from the following equation:

$$h = \left(\frac{2\pi N}{60}\right)^2 \frac{2\pi\mu}{E_f}\left(\frac{R^4}{4} - \frac{R_0^4}{4}\right) \qquad (13)$$

Other parameters required are: $W_s = 101000$ lb (45804.99 Kg), $P_{max} = 1000$ psi (6.89655×10^6 Pa), $\Delta T_{max} = 50\,^{\circ}$F (10 °C), $h_{min} = 0.001$ in (0.00254 cm), $g = 386.4$ in/seg^2 (981.456 cm/seg^2), and $N = 750$.

3.2. Example 2 : Himmelblau's Nonlinear Optimization Problem

This problem was originally proposed by Himmelblau [7], and it was chosen to try the new approach because it has been used before as a benchmark for GA-based techniques that use penalties. In this problem, there are 5 design variables $(x_1, x_2, x_3, x_4, x_5)$, 6 nonlinear inequality constraints and 10 boundary conditions. The problem can be stated as follows:

$$Min \quad f(\mathbf{X}) = 5.3578547x_3^2 + 0.8356891x_1x_5 \qquad (14)$$

Subject to:

$$g_1(\mathbf{X}) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5 \qquad (15)$$

$$g_2(\mathbf{X}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \qquad (16)$$

$$g_3(\mathbf{X}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \qquad (17)$$

$$0 \le g_1(\mathbf{X}) \le 92 \qquad (18)$$

$$90 \le g_2(\mathbf{X}) \le 110 \qquad (19)$$

$$20 \le g_3(\mathbf{X}) \le 25 \qquad (20)$$

$$78 \le x_1 \le 102 \qquad (21)$$

$$33 \le x_2 \le 45 \qquad (22)$$

$$27 \le x_3 \le 45 \qquad (23)$$

$$27 \le x_4 \le 45 \qquad (24)$$

$$27 \le x_5 \le 45 \qquad (25)$$

4. Comparison of Results

4.1. Example 1

This problem was solved before by Deb and Goyal [8] using GeneAS (Genetic Adaptive Search, which is a real-coded GA) and a traditional (binary) genetic algorithm (both approaches used a penalty function), and by Siddall [9] using ADRANS (Gall's adaptive random search with a penalty function). Their results were compared against those produced by the approach proposed in this paper, and are shown in Table 1. Notice that the solution reported by Siddall [10] is infeasible (it violates the first constraint). The solution shown for the technique proposed here is the best produced after 81 runs in which the crossover and mutation rates were iterated from 0.1 to 0.9 (at increments of 0.1) in a nested loop in which the mutation rate was iterated first while the crossover rate remained fixed. For example, given an initial crossover rate of 0.1, the nine values from 0.1 to 0.9 were used for the mutation rate, running a GA for each pair of parameters (crossover and mutation rates). Once the loop for the mutation rate finished, the crossover rate was incremented by 0.1 and the loop for the mutation rate was run again from 0.1 to 0.9. The process was repeated until the crossover rate reached 0.9. The following ranges were used for the design variables: $1.000 \le x_1 \le 16.000$, $1.000 \le x_2 \le 16.000$, $1.0 \times 10^{-6} \le x_3 \le 16.0 \times 10^{-6}$, $1.000 \le x_4 \le 16.000$. The values of all variables were considered with a 3-decimal precision (the values of x_3 were considered as multiples of 1.0×10^{-6}). The total population size used was 160 (20 individuals for each of the 8 sub-populations) and the maximum number of generations was 100.

4.2. Example 2

This problem was originally proposed by Himmelblau [7] and solved using the Generalized Reduced Gradient method (GRG). Gen and Cheng [12] solved this problem using a genetic algorithm based on both local and global reference. The result shown in Table 2 is the best of the two reported by Gen and Cheng [12]. Homaifar, Qi, and Lai [11] solved this problem using

Design Variables	Best solution found			
	This paper	GeneAS	BGA	Siddall
$x_1\,(R)$	6.271	6.778	7.077	7.155
$x_2\,(R_0)$	12.901	6.234	6.549	6.689
$x_3\,(\mu) \times 10^{-6}$	5.605	6.096	6.619	8.321
$x_4\,(Q)$	2.938	3.809	4.849	9.168
$g_1\,(\mathbf{X})$	2126.86734	8329.7681	1440.6013	-11086.7430
$g_2\,(\mathbf{X})$	68.0396	177.3527	297.1495	402.4493
$g_3\,(\mathbf{X})$	3.705191	10.684543	17.353800	35.057196
$g_4\,(\mathbf{X})$	0.000559	0.000652	0.000891	0.001542
$g_5\,(\mathbf{X})$	0.666000	0.544000	0.528000	0.466000
$g_6\,(\mathbf{X})$	0.000805	0.000717	0.000624	0.000144
$g_7\,(\mathbf{X})$	849.718683	83.618221	467.686527	563.644401
$f(\mathbf{X})$	**1950.2860**	**2161.4215**	**2296.2119**	**2288.2268**

Table 1
Comparison of results for the first example (design of a hydrostatic thrust bearing).

a genetic algorithm with a population size of 400, and their results were the best previously reported in the literature for this problem (see Table 2). The solution shown for the technique proposed here is the best produced after 81 runs in which the crossover and mutation rates were iterated from 0.1 to 0.9 in a nested loop, and the following ranges were used for the design variables: $78.0000 \leq x_1 \leq 102.0000$, $33.0000 \leq x_2 \leq 45.0000$, $27.0000 \leq x_3 \leq 45.0000$, $27.0000 \leq x_4 \leq 45.0000$, and $27.0000 \leq x_5 \leq 45.0000$. The values for all the variables were considered with a 4-decimal precision. The total population size used was 160 (40 individuals for each of the 4 sub-populations) and the maximum number of generations was 100.

5. Discussion

In the examples presented before, the new approach found better solutions than those previously reported in the literature by using relatively small sub-population sizes. However, the selection of an appropriate sub-population size (assuming that they are all the same) remains an issue as when using a GA with a single population. Determining the maximum number of generations presents a similar problem, although in this case it is possible to monitor the population so that the GA is stopped when there is not enough diversity anymore (e.g., when the fitness of the best individual is very close to the mean fitness of the population). The main drawback of the new technique may be the number of sub-populations that may be needed in larger problems, since they will increase linearly with the number of constraints. However, it is possible to deal with that problem in two different ways: first, some constraints could be tied; that means that two or more constraints could be assigned to the same sub-population. That would significantly reduce the number of sub-populations in highly constrained problems. Second, we could parallelize the approach, in which case a high number of sub-populations will not be a serious drawback, since they could be processed concurrently. The current algorithm would however need modifications as to decide the sort of interactions between a master process (responsible for actually optimizing the whole problem) and the slave sub-processes (all the sub-populations responsible for the constraints of the problem). That is in fact the area of research currently being pursued by the author.

6. Conclusions and Future Work

This paper has introduced a new GA-based approach that uses a multiobjective optimization technique to handle constraints, instead of using the more traditional penalty approach. The new approach worked well in some test problems that had been pre-

Design Variables	Best solution found			
	This paper	Gen	Homaifar	GRG
x_1	78.5958	81.4900	78.0000	78.6200
x_2	33.0100	34.0900	33.0000	33.4400
x_3	27.6460	31.2400	29.9950	31.0700
x_4	45.0000	42.2000	45.0000	44.1800
x_5	45.0000	34.3700	36.7760	35.2200
$g_1(\mathbf{X})$	91.956402	90.522543	90.714681	90.520761
$g_2(\mathbf{X})$	100.545111	99.318806	98.840511	98.892933
$g_3(\mathbf{X})$	20.251919	20.060410	19.999935	20.131578
$f(\mathbf{X})$	**−30810.359**	**−30183.576**	**−30665.609**	**−30373.949**

Table 2
Comparison of results for the fourth example (Himmelblau's function).

viously solved using GA-based and mathematical programming techniques, producing results better than those previously reported in the literature. The technique was able to achieve such good results with relatively small sub-populations, and without the need to use any extra parameters for the GA, although the issue of selecting the most appropriate sub-population size as well as the maximum number of generations, remains open as in the case of the simple GA [1]. The first extension of this work will be to develop a parallel implementation of the algorithm. Some interesting issues that a parallel version of this algorithm arise are for example the migration policies required to exchange information, the consequences of restricting crossover, the effect of the topology used by the parallel architecture on the overall performance of the GA, the significance of the evolution of the small sub-populations responsible for the constraints concurrently with the evolution of a main population containing a mixed of feasible and infeasible solutions.

7. Acknowledgments

The author acknowledges support from CONACyT through project number $I - 29870A$.

References

[1] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts.

[2] Richardson, J. T., Palmer, M. R., Liepins, G., and Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 191–197. Morgan Kaufmann Publishers, George Mason University.

[3] Surry, P. D., Radcliffe, N. J., and Boyd, I. D. (1995). A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method. In T. C. Fogarty, editor, *Evolutionary Computing. AISB Workshop. Selected Papers*, Lecture Notes in Computer Science, pp. 166–180. Springer-Verlag, Sheffield, U.K.

[4] Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 93–100. Lawrence Erlbaum.

[5] Parmee, I. C. and Purchase, G. (1994). The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pp. 97–102. University of Plymouth, UK.

[6] Coello, Carlos A. (1999). Treating constraints as objectives for single-objective optimization. *Computers in Industry*, (accepted for publication).

[7] Himmelblau, D. M. (1972). *Applied Nonlinear Programming*. McGraw-Hill, New York.

[8] Deb, K. and Goyal, M. (July 1995). Optimizing Engineering Designs Using a Combined Genetic Search. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 521–528. Morgan Kauffman Publishers, San Mateo, California.

[9] Siddall, J. N. (1982). *Optimal Engineering Design. Principles and Applications*. Marcel Dekker, New York.

[10] Siddall, J. N. (1972). *Analytical Design-Making in Engineering Design*. Prentice-Hall.

[11] Homaifar, A., Lai, S. H. Y., and Qi, X. (1994). Con-

strained Optimization via Genetic Algorithms. *Simulation*, 62(4):242–254.

[12] Gen, M. and Cheng, R. (1997). *Genetic Algorithms & Engineering Design*. John Wiley & Sons, Inc, New York.