Self-Adaptive Penalties for GA-based Optimization

Carlos A. Coello Coello
Laboratorio Nacional de Informática Avanzada
Rébsamen 80, Xalapa, Veracruz 91090, México
ccoello@xalapa.lania.mx

Abstract- This paper introduces the notion of using co-evolution to adapt the penalty factors of a fitness function incorporated in a genetic algorithm for numerical optimization. The proposed approach produces solutions even better than those previously reported in the literature for other (GA-based and mathematical programming) techniques that have been particularly fine-tuned using a normally lengthy trial and error process to solve a certain problem or set of problems. The present technique is also easy to implement and suitable for parallelization, which is a necessary further step to improve its current performance.

1 Introduction

The importance of genetic algorithms (GAs) as a powerful tool for engineering optimization has been widely shown in the last few years through a vast amount of applications ([1, 2]). However, even when GAs have been successful in many practical applications, the quality of the solutions that they produce rely not only on the stochastic nature of the technique, but also on the way in which the objective function is converted to a "fitness function" that can "guide" the GA to the desired region of the search space.

One of the key problems for using GAs in practical applications is how to design the fitness function. A comparative estimate of how good is a solution turns out to be enough in most cases (e.g., the largest value has to be closer to the global maximum if we are trying to maximize the objective function), but if we are dealing with constrained problems, we have to find a way of estimating also how close is an infeasible solution from the feasible region. This is not an easy task, since most real-world problems have complex linear and non-linear constraints, and several approaches have been proposed in the past to handle them [3, 4, 5, 6]. From those, the penalty function seems to be yet the most popular technique for engineering problems, but the intrinsic difficulties to define good penalty values makes even harder the optimization process using a GA [7]. In this paper, a technique based on the concept of co-evolution is used to create two populations that interact with each other in such a way that one population evolves the penalty factors to be used by the fitness function of the main population, which is responsible for optimizing the objective function. The approach has been tested with several single-objective optimization problems with linear and non-linear inequality constraints and its results are compared with those produced by other (GA-based and mathematical programming) approaches reported in the literature.

2 Use of Self-Adaptive Penalties

Michalewicz et al. [4, 5] have recognized the importance of using adaptive penalties in evolutionary optimization, and considered this approach as a very promising direction of research on evolutionary optimization. The technique proposed in this paper aims to implement this idea using the concept of co-evolution, under which two (or more) populations are evolved either concurrently or interactively, and such populations exchange information in the process. Paredis [8] has used co-evolution for constraint satisfaction (combinatorial optimization) problems, but not for numerical optimization. In his approach, a population of potential solutions co-evolves with a population of constraints: fitter solutions satisfy more constraints, whereas fitter constraints are violated by more solutions.

The approach introduced in this paper uses a conventional penalty function [1, 7] rather than trying to handle constraints in an entirely different way (see for example [6, 5]). The reason is that penalty functions are still the most popular approach to handle constraints in practical applications [5, 6], whereas the newer approaches have normally been used only to deal with very specific (and generally unrealistic) problems.

The problem that we want to solve is:

$$\text{Optimize} \quad f(\mathbf{X}) \tag{1}$$

Subject to :

$$g_i(\mathbf{X}) \leq 0 \quad i = 1, \ldots, p \tag{2}$$

Only inequality constraints are considered in this paper, since penalty functions are not very suitable to handle equality constraints as hard constraints, and there are other approaches which are more suitable to handle them [9].

In previous applications, a penalty function that included information about both the number of constraints violated and the degree of violation of each, has been found very effective by a number of researchers [10, 6] to guide the genetic algorithm to (at least near) optimal solutions. The expression used to compute the fitness value of an individual for the purposes of this paper is (assuming maximization):

$$fitness_i = f_i(\mathbf{X}) - (coef \times w_1 + viol \times w_2) \tag{3}$$

where $f_i(\mathbf{X})$ is the value of the objective function for the given set of variable values encoded in the chromosome i; w_1 and w_2 are 2 penalty factors (considered as integers in this paper); $coef$ is the sum of all the amounts by which the

constraints are violated:

$$coef = \sum_{i=1}^{p} g_i(\mathbf{X}) \quad \forall g_i(\mathbf{X}) > 0 \tag{4}$$

$viol$ is an integer factor, initialized to zero and incremented by one for each constraint of the problem that is violated, regardless of the amount of violation (i.e., we only count the number of constraints violated but not the magnitude in which each constraint is violated).

According to this approach, the penalty is actually split into two values ($coef$ and $viol$), so that the GA has enough information not only about how many constraints were violated, but also about the amounts in which such constraints were violated. This follows Richardson's suggestion [7] about using penalties that are guided by the distance to feasibility.

We will assume that we have 2 different populations $P1$ and $P2$ with corresponding sizes $M1$ and $M2$. The second of these populations ($P2$) encodes the set of weight combinations (w_1 and w_2) that will be used to compute the fitness value of the individuals in $P1$ (i.e., $P2$ contains the penalty factors that will be used in the fitness function). The idea is to use one population to evolve solutions (as in a conventional genetic algorithm), and another to evolve the penalty factors w_1 and w_2. A graphical representation of this approach may be seen in Figure 1. Notice that for each individual A_j in $P2$ there is an instance of $P1$. However, the population $P1$ is reused for each new element A_j processed from $P2$. Each individual A_j ($1 \leq j \leq M2$) in $P2$ is decoded and the weight combination produced (i.e., the penalty factors) is used to evolve $P1$ during a certain number ($Gmax1$) of generations. The fitness of each individual B_k ($1 \leq k \leq M1$) is computed using equation (3), keeping the penalty factors constant for every individual in the instance of $P1$ corresponding to the individual A_j being processed. After evolving each $P1$ corresponding to every A_j in $P2$ (there is only one instance of $P1$ for each individual in $P2$), we compute the best average fitness produced using:

$$average\_fitness_j = \sum_{i=1}^{M1} \left(\frac{fitness_i}{count\_feasible} \right) + count\_feasible \ \forall \mathbf{X} \in \mathcal{F} \tag{5}$$

In equation (5), we add the fitnesses of all feasible solutions in $P1$, and obtain an average of them (the integer variable $count\_feasible$ is a counter that indicates how many feasible solutions were found in the population). Notice that although the summation ranges over all the individuals in $P1$, only feasible solutions are considered. The reason for this is that if we do not exclude infeasible solutions from this computation, the selection mechanism of the GA may bias the population towards regions of the search space where there are solutions with a very low weight combination (w_1 and w_2). Such solutions may have good fitness values, and still be infeasible. The reason for that is that low values of w_1 and w_2 may produce penalties that are not big enough to outweight the value of the objective function. Notice also the use of $count\_feasible$ to avoid stagnation at certain regions in which only very few individuals will have a good fitness or will be even feasible. By adding this quantity to the average fitness of the feasible individuals in the population, we will be encouraging the GA to move towards regions in which lie not only feasible solutions with good fitness values, but there are also a lot of them. In practice, it may be necessary to apply a scaling factor to the average of the fitness before adding $count\_feasible$, to avoid that the GA gets trapped in local optima, and we do in fact use a scaling factor in the experiments reported here. However, such scaling factor is not very difficult to compute because we are assuming populations of constant size (such size must be defined before running the GA), and the range of the fitness values can be easily obtained at each generation, because we know the maximum and minimum fitness values in the population at each generation. Equation (5) may, of course, limit the applicability of the proposed approach to situations in which there is at least one single fully feasible solution in the first generation, but notice that since there are several combinations of weights considered for each instance of $P1$, we can afford having this situation and still be able to find a solution to the problem at hand. In fact, in our experiments, we had sometimes this situation (i.e., no feasible solution for an instance of $P1$ was available) and the GA was still able to move towards the optimum.

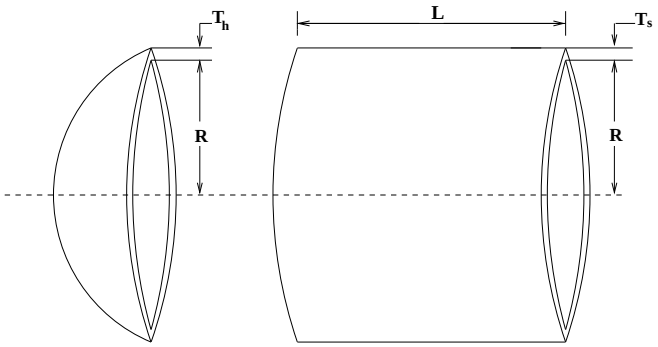

Figure 2: Center and end section of the pressure vessel used for the first example.

The process indicated above is repeated until all individuals in $P2$ have a fitness value (the best $average\_fitness$ of their corresponding $P1$). Then, $P2$ is evolved one generation using conventional genetic operators (i.e., crossover and mutation) and the new $P2$ produced is used to start the same process all over again. It is important to notice that the interaction between $P1$ and $P2$ introduces diversity in both populations, which keeps the GA from easily converging to a local optimum.

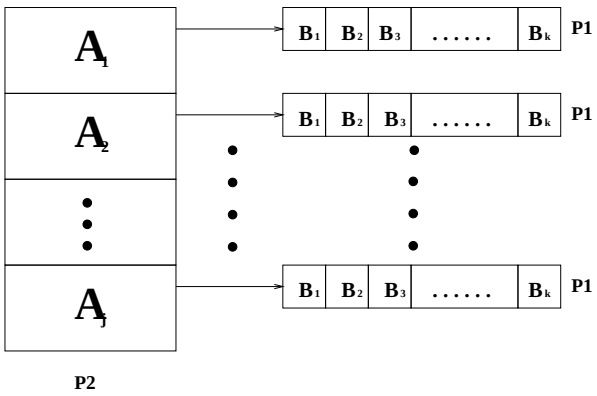

Figure 1: Graphical representation of the GA-based approach to handle constraints proposed in this paper.

3 Examples

Several examples taken from the optimization literature will be used to show the way in which the proposed approach works. These examples have linear and nonlinear constraints, and have been previously solved using a variety of other techniques (both GA-based and traditional mathematical programming methods), which is useful to determine the quality of the solutions produced by the proposed approach.

3.1 Example 1 : Design of a Pressure Vessel

A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure 2. The objective is to minimize the total cost, including the cost of the material, forming and welding. There are four design variables: T_s (thickness of the shell), T_h (thickness of the head), R (inner radius) and L (length of the cylindrical section of the vessel, not including the head). T_s and T_h are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates, and R and L are continuous. Using the same notation given by Kannan and Kramer [11], the problem can be stated as follows:

Minimize :

$$F(\mathbf{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + \\ +3.1661x_1^2x_4 + 19.84x_1^2x_3 \quad (6)$$

Subject to :

$$g_1(\mathbf{X}) = -x_1 + 0.0193x_3 \leq 0 \quad (7)$$

$$g_2(\mathbf{X}) = -x_2 + 0.00954x_3 \leq 0 \quad (8)$$

$$g_3(\mathbf{X}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0 \quad (9)$$

$$g_4(\mathbf{X}) = x_4 - 240 \leq 0 \quad (10)$$

3.2 Example 2 : Welded Beam Design

A welded beam is designed for minimum cost subject to constraints on shear stress (τ), bending stress in the beam (σ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints [12]. There are four design variables as shown in Figure 3 [12]: h (x_1), l (x_2), t (x_3) and b (x_4).

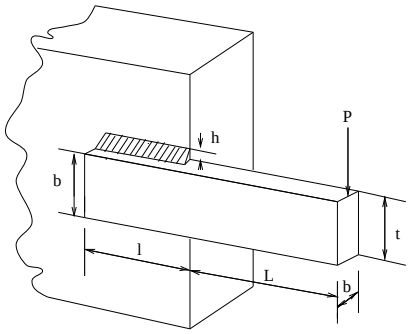

Figure 3: The welded beam used for the second example.

The problem can be stated as follows:

Minimize:

$$F(\mathbf{X}) = 1.10471x_1^2x_2 + \\ +0.04811x_3x_4(14.0 + x_2) \quad (11)$$

Subject to :

$$g_1(\mathbf{X}) = \tau(\mathbf{X}) - \tau_{max} \leq 0 \quad (12)$$

$$g_2(\mathbf{X}) = \sigma(\mathbf{X}) - \sigma_{max} \leq 0 \quad (13)$$

$$g_3(\mathbf{X}) = x_1 - x_4 \leq 0 \quad (14)$$

$$g_4(\mathbf{X}) = 0.10471x_1^2 + \\ +0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \quad (15)$$

$$g_5(\mathbf{X}) = 0.125 - x_1 \leq 0 \quad (16)$$

$$g_6(\mathbf{X}) = \delta(\mathbf{X}) - \delta_{max} \leq 0 \quad (17)$$

$$g_7(\mathbf{X}) = P - P_c(\mathbf{X}) \leq 0 \quad (18)$$

where

$$\tau(\mathbf{X}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \quad (19)$$

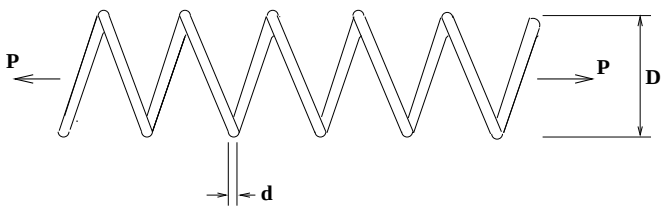

Figure 4: Tension/compression string used for the third example.

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right) \quad (20)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \quad (21)$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \quad (22)$$

$$\sigma(\mathbf{X}) = \frac{6PL}{x_4x_3^2}, \delta(\mathbf{X}) = \frac{4PL^3}{Ex_3^3x_4} \quad (23)$$

$$P_c(\mathbf{X}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \quad (24)$$

$$P = 6000\,lb,\ \ L = 14\,in,\ \ \delta_{max} = 0.25\,in$$
$$E = 30 \times 10^6\,psi,\ \ G = 12 \times 10^6\,psi$$
$$\tau_{max} = 13,600\,psi,\ \ \sigma_{max} = 30,000\,psi$$

3.3 Example 3 : Minimization of the Weight of a Tension/Compression String

This problem was described by Arora [13] and Belegundu [14], and it consists of minimizing the weight of a tension/compression spring (see Figure 4) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the mean coil diameter D, the wire diameter d and the number of active coils N.

Formally, the problem can be expressed as:

$$\text{Minimize} \quad (N+2)Dd^2 \quad (25)$$

Subject to

$$g_1(\mathbf{X}) = 1 - \frac{D^3N}{71785d^4} \leq 0 \quad (26)$$

$$g_2(\mathbf{X}) = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \leq 0 \quad (27)$$

$$g_3(\mathbf{X}) = 1 - \frac{140.45d}{D^2N} \leq 0 \quad (28)$$

$$g_4(\mathbf{X}) = \frac{D+d}{1.5} - 1 \leq 0 \quad (29)$$

Parameter	Value
Pop_size$_1$	60
Pop_size$_2$	30
Gmax$_1$	25
Gmax$_2$	20

Table 1: Parameters of the GA used to solve all the examples.

3.4 Example 4 : Himmelblau's Nonlinear Optimization Problem

This problem was originally proposed by Himmelblau [15], and it was chosen to try the approach proposed here because it has been used before as a benchmark for several other GA-based techniques that use penalties [16]. In this problem, there are five design variables (x_1, x_2, x_3, x_4, x_5), 6 nonlinear inequality constraints and ten boundary conditions. The problem can be stated as follows:

$$\text{Minimize} \ \ f(\mathbf{X}) = 5.3578547x_3^2 + \\ +0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \quad (30)$$

Subject to:

$$g_1(\mathbf{X}) = 85.334407 + 0.0056858x_2x_5 + \\ 0.00026x_1x_4 - 0.0022053x_3x_5 \quad (31)$$

$$g_2(\mathbf{X}) = 80.51249 + 0.0071317x_2x_5 + \\ 0.0029955x_1x_2 + 0.0021813x_3^2 \quad (32)$$

$$g_3(\mathbf{X}) = 9.300961 + 0.0047026x_3x_5 + \\ 0.0012547x_1x_3 + 0.0019085x_3x_4 \quad (33)$$

$$0 \leq g_1(\mathbf{X}) \leq 92 \quad (34)$$

$$90 \leq g_2(\mathbf{X}) \leq 110 \quad (35)$$

$$20 \leq g_3(\mathbf{X}) \leq 25 \quad (36)$$

$$78 \leq x_1 \leq 102 \quad (37)$$

$$33 \leq x_2 \leq 45 \quad (38)$$

$$27 \leq x_3 \leq 45 \quad (39)$$

$$27 \leq x_4 \leq 45 \quad (40)$$

$$27 \leq x_5 \leq 45 \quad (41)$$

4 Comparison of Results

To make a fair comparison, all the following examples were solved using the same set of parameters shown in Table 1. We used a GA with fixed-point representation [17, 10], according to which a chromosome is a string of the form $\langle d_1, d_2, \ldots, d_m \rangle$, where $d_1, d_2, \ldots, d_m$ are digits (numbers between zero and nine). In previous work, the author has

shown that this representation compares favorably to its binary counterpart in numerical optimization problems [18, 17, 10]. We also used uniform crossover with a crossover probability of 0.8 and non-uniform mutation [9] with an initial mutation rate of 0.1 to allow a high exploratory behavior of the GA at earlier generations, and focus more the search into certain regions as the GA reached its last generations.

4.1 Example 1

This problem was solved before by Deb [19] using GeneAS (Genetic Adaptive Search), by Kannan and Kramer using an augmented Lagrangian Multiplier approach [11], by Fu et al. [20] using Integer-Discrete-Continuous Nonlinear Programming, and by Cao and Wu [21] using Evolutionary Programming. Their results were compared against those produced by the approach proposed in this paper, and are shown in Table 2. The solution shown for the technique proposed here is the best produced after 11 runs, and using the following ranges for the design variables and the weights: $1 \leq x_1 \leq 99$, $1 \leq x_2 \leq 99$, $10.0000 \leq x_3 \leq 200.0000$, $10.0000 \leq x_4 \leq 200.0000$, $1 \leq w_1 \leq 999$, and $1 \leq w_2 \leq 999$. The values for x_1 and x_2 were considered as integer multiples of 0.0625, the weights w_1 and w_2 were considered as integers, and the values of x_3 and x_4 were considered with a 4-decimals precision. The mean from the 11 runs performed was $f(\mathbf{X}) = 6293.84323196$, with a standard deviation of 7.41328537. The worst solution found was $f(\mathbf{X}) = 6308.14965192$, which is better than any of the solutions previously reported in the literature. The solution at the median was $f(\mathbf{X}) = 6290.01873568$ (corresponding to $x_1 = 0.8125$, $x_2 = 0.4372$, $x_3 = 40.3302$ and $x_4 = 200.0000$), which is still about 2% better than the best solution previously reported.

4.2 Example 2

This problem was solved before by Deb [22] using a simple genetic algorithm with binary representation, and a traditional penalty function as suggested by Goldberg [1], and by Ragsdell and Phillips [23] using geometric programming. Ragsdell and Phillips also compared their results with those produced by the methods contained in a software package called "Opti-Sep" [24], which includes the following numerical optimization techniques: ADRANS (Gall's adaptive random search with a penalty function), APPROX (Griffith and Stewart's successive linear approximation), DAVID (Davidon-Fletcher-Powell with a penalty function), MEMGRD (Miele's memory gradient with a penalty function), SEEK1 & SEEK2 (Hooke and Jeeves with 2 different penalty functions), SIMPLX (Simplex method with a penalty function) and RANDOM (Richardson's random method). Their results were compared against those produced by the approach proposed in this paper, and are shown in Table 3. In the case of Siddall's techniques [24], only the best solution produced by the techniques contained in "Opti-Sep" is displayed. The solution shown for the technique proposed here is the best produced after 11 runs, and using the following ranges for the design variables and the weights: $0.1000 \leq x_1 \leq 2.0000$, $0.1000 \leq x_2 \leq 10.0000$, $0.1000 \leq x_3 \leq 10.0000$, $0.1000 \leq x_4 \leq 2.0000$, $1 \leq w_1 \leq 999$, and $1 \leq w_2 \leq 999$. The values for x_1 to x_4 were considered with a 4-decimals precision, and the weights w_1 and w_2 were considered as integers. The mean from the 11 runs performed was $f(\mathbf{X}) = 1.77197269$, with a standard deviation of 0.01122281. The worst solution found was $f(\mathbf{X}) = 1.7858346524$, which is better than any of the solutions produced by any of the other techniques depicted in Table 3. The solution at the median was $f(\mathbf{X}) = 1.77358615$ (corresponding to $x_1 = 0.1996$, $x_2 = 3.6428$, $x_3 = 9.0507$ and $x_4 = 0.2100$), which is about 27% better than the best solution previously reported.

4.3 Example 3

This problem was solved before by Belegundu [14] using eight numerical optimization techniques (CONMIN, OPTDYN, LINMR, GRP-UI, SUMT, M-3, M4, and M-5). Only the best feasible result reported by him is shown in Table 4. Additionally, Arora [13] also solved this problem using a numerical optimization technique called Constraint Correction at constant Cost (CCC). In the experiments reported here, the GA handled all constraints are hard, so that the solutions produced were considered valid only if all of them were fully satisfied. Nevertheless, the proposed approach was able to find a better (feasible) solution than Arora's technique (whose solution slightly violates one constraint), as can be seen in Table 4. The solution shown for the technique proposed here is the best produced after 11 runs, and using the following ranges for the design variables and the weights: $0.050000 \leq x_1 \leq 2.000000$, $0.250000 \leq x_2 \leq 1.300000$, $2.000000 \leq x_3 \leq 15.000000$, $1 \leq w_1 \leq 999$, and $1 \leq w_2 \leq 999$. The values for x_1 to x_4 were considered with a 6-decimals precision, and the weights w_1 and w_2 were considered as integers. The mean from the 11 runs performed was $f(\mathbf{X}) = 0.01276920$, with a standard deviation of 3.939×10^{-5}. The worst solution found was $f(\mathbf{X}) = 0.0128220825$, which is better than Belegundu's result. The solution at the median was $f(\mathbf{X}) = 0.0127557615$ (corresponding to $x_1 = 0.051461$, $x_2 = 0.351022$, and $x_3 = 11.721943$), which is better than the best feasible solution previously reported.

4.4 Example 4

This problem was originally proposed by Himmelblau [15] and solved using the Generalized Reduced Gradient method (GRG). Gen and Cheng [16] solved this problem using a genetic algorithm based on both local and global reference. The result shown in Table 5 is the best found with their approach. Homaifar, Qi, and Lai [25] solved this problem using a genetic algorithm with a population size of 400, and their re-

Design Variables	Best solution found				
	This paper	**GeneAS [19]**	**Kannan [11]**	**Fu [20]**	**Cao [21]**
$x_1(T_s)$	0.8125	0.9375	1.125	1.125	1.000
$x_2(T_h)$	0.4375	0.5000	0.625	0.625	0.625
$x_3(R)$	40.3239	48.3290	58.291	48.3807	51.1958
$x_4(L)$	200.0000	112.6790	43.690	111.7449	90.7821
$g_1(\mathbf{X})$	-0.034324	-0.004750	0.000016	-0.191252	-0.011921
$g_2(\mathbf{X})$	-0.052847	-0.038941	-0.068904	-0.163448	-0.136592
$g_3(\mathbf{X})$	-27.105845	-3652.876838	-21.220104	-72.970137	-13584.583968
$g_4(\mathbf{X})$	-40.00000	-127.321000	-196.310000	-128.25510	-149.2179
$f(\mathbf{X})$	**6288.7445**	**6410.3811**	**7198.0428**	**8049.3411**	**7108.6160**

Table 2: Comparison of the results for the first example (optimization of a pressure vessel).

Design Variables	Best solution found			
	This paper	**Deb [22]**	**Siddall [24]**	**Ragsdell [23]**
$x_1(h)$	0.2088	0.2489	0.2444	0.2455
$x_2(l)$	3.4205	6.1730	6.2189	6.1960
$x_3(t)$	8.9975	8.1789	8.2915	8.2730
$x_4(b)$	0.2100	-0.2533	0.2444	0.2455
$g_1(\mathbf{X})$	-0.337812	-5758.603777	-5743.502027	-5743.826517
$g_2(\mathbf{X})$	-353.902604	-255.576901	-4.015209	-4.715097
$g_3(\mathbf{X})$	-0.00120	-0.004400	0.000000	0.000000
$g_4(\mathbf{X})$	-3.411865	-2.982866	-3.022561	-3.020289
$g_5(\mathbf{X})$	-0.08380	-0.123900	-0.119400	-0.120500
$g_6(\mathbf{X})$	-0.235649	-0.234160	-0.234243	-0.234208
$g_7(\mathbf{X})$	-363.232384	-4465.270928	-3490.469418	-3604.275002
$f(\mathbf{X})$	**1.74830941**	**2.43311600**	**2.38154338**	**2.38593732**

Table 3: Comparison of the results for the second example (optimal design of a welded beam).

Design Variables	Best solution found		
	This paper	**Arora [13]**	**Belegundu [14]**
$x_1(d)$	0.051480	0.053396	0.050000
$x_2(D)$	0.351661	0.399180	0.315900
$x_3(N)$	11.632201	9.185400	14.25000
$g_1(\mathbf{X})$	-0.002080	0.000019	-0.000014
$g_2(\mathbf{X})$	-0.000110	-0.000018	-0.003782
$g_3(\mathbf{X})$	-4.026318	-4.123832	-3.938302
$g_4(\mathbf{X})$	-4.026318	-0.698283	-0.756067
$f(\mathbf{X})$	**0.0127047834**	**0.0127302737**	**0.0128334375**

Table 4: Comparison of the results for the third example (minimization of the weight of a tension/compression spring).

sults were previously the best reported in the literature (see Table 5). The solution shown for the technique proposed here is the best produced after 11 runs, and using the following ranges for the design variables and the weights: $78.0000 \leq x_1 \leq 102.0000$, $33.0000 \leq x_2 \leq 45.0000$, $27.0000 \leq x_3 \leq 45.0000$, $27.0000 \leq x_4 \leq 45.0000$, $27.0000 \leq x_5 \leq 45.0000$, $1 \leq w_1 \leq 999$, and $1 \leq w_2 \leq 999$. The values for x_1 to x_5 were considered with a 4-decimals precision, and the weights w_1 and w_2 were considered as integers. The mean from the 11 runs performed was $f(\mathbf{X}) = -30984.24070309$, with a standard deviation of 73.63353661.The worst solution found was $f(\mathbf{X}) = -30792.4077377525$, which is better than the best solution previously reported in the literature. The solution at the median was $f(\mathbf{X}) = -31017.21369099$ (corresponding to $x_1 = 78.010$, $x_2 = 33.030$, $x_3 = 27.119$, $x_4 = 45.000$, and $x_5 = 44.872$).

5 Discussion

Despite the fact that the proposed approach requires more function evaluations than running a GA on a single population, it could be argued that in practice the proposed approach turns out to be more efficient because it does not require the traditional fine-tuning of a simple GA which is normally performed by trial and error and normally takes a considerable amount of time.

The set of parameters proposed at the beginning of this paper were derived based on a series of experiments. Initially, it was found that in most cases a fairly small population size for $P2$ (≤ 40 chromosomes) would suffice to find reasonable solutions (unless within a 5% vicinity of the best solution known), but the size of $P1$ was much more dependent on the nature of the problem, although in all the cases reported in this paper it was sufficient to use relatively small sizes (between 30 and 60 chromosomes), although the largest value was chosen to allow for a fair comparison with other techniques. Similarly, the effect that the maximum number of generations produced in the results seemed to be more significant for $P1$ than for $P2$. This is not very surprising, since $P1$ is really the population responsible for performing the optimization. It is interesting to mention that in the experiments performed, it was found that the increment of the maximum number of generations for $P1$ would normally improve the quality of the solution, but there was always a threshold after which an increment did not affect the results in a significant manner. On the other hand, the increment of the maximum number of generations for $P2$ was normally not very beneficial, and that was the reason why it was normally preferred to use smaller values for $Gmax2$ than for $Gmax1$.

6 Conclusions and Future Work

This paper has proposed a GA-based technique that uses co-evolution to adjust automatically the weight factors of a penalty function to find the optimum of a constrained optimization problem. Due to the intrinsic limitations of penalty functions to handle equality constraints, only inequality constraints were considered in this work, although alternative hybrid approaches [4] may be used in combination with the proposed technique in order to deal with equality constraints, too. The new technique worked well in several test problems that had been previously solved using GA-based and mathematical programming techniques, producing in all cases results better than those previously reported in the literature. However, performance issues remain to be solved since the total number of fitness function evaluations in all the experiments reported in this paper can be considered high ($900,000$), and it is desirable to develop a parallel version of this algorithm in the future. Also, we expect to compare this approach to other evolutionary-based constraint-handling techniques and to experiment with a simpler approach in which a hillclimber modifies the weights of the penalty function instead of using another GA for that task.

Acknowledgments

The author would like to thank the anonymous reviewers for their comments and acknowledges the support received from CONACyT through project number I-29870 A.

Bibliography

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

[2] Thomas Bäck, editor. *Proceedings of the Seventh International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers, San Mateo, California, July 1997.

[3] Zbigniew Michalewicz. Genetic Algorithms, Numerical Optimization, and Constraints. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 151–158, San Mateo, California, July 1995. University of Pittsburgh, Morgan Kaufmann Publishers.

[4] Zbigniew Michalewicz, Dipankar Dasgupta, R. Le Riche, and Marc Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30(4):851–870, September 1996.

[5] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[6] Dipankar Dasgupta and Zbigniew Michalewicz, editors. *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, Berlin, 1997.

Design Variables	Best solution found			
	This paper	**Gen [16]**	**Homaifar [25]**	**GRG [15]**
x_1	78.0495	81.4900	78.0000	78.6200
x_2	33.0070	34.0900	33.0000	33.4400
x_3	27.0810	31.2400	29.9950	31.0700
x_4	45.0000	42.2000	45.0000	44.1800
x_5	44.9400	34.3700	36.7760	35.2200
$g_1(\mathbf{X})$	91.997635	90.522543	90.714681	90.520761
$g_2(\mathbf{X})$	100.407857	99.318806	98.840511	98.892933
$g_3(\mathbf{X})$	20.001911	20.060410	19.999935	20.131578
$f(\mathbf{X})$	$\mathbf{-31020.859}$	$\mathbf{-30183.576}$	$\mathbf{-30665.609}$	$\mathbf{-30373.949}$

Table 5: Comparison of the results for the fourth example (Himmelblau's function).

[7] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some guidelines for genetic algorithms with penalty functions. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197, George Mason University, 1989. Morgan Kaufmann Publishers.

[8] J. Paredis. Co-evolutionary Constraint Satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, pages 46–55, New York, 1994. Springer Verlag.

[9] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, second edition, 1992.

[10] Carlos A. Coello Coello and Alan D. Christiansen. A simple genetic algorithm for the design of reinforced concrete beams. *Engineering with Computers*, 13(4):185–196, 1997.

[11] B. K. Kannan and S. N. Kramer. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design. *Journal of Mechanical Design. Transactions of the ASME*, 116:318–320, 1994.

[12] Singiresu S. Rao. *Engineering Optimization*. John Wiley and Sons, third edition, 1996.

[13] Jasbir S. Arora. *Introduction to Optimum Design*. McGraw-Hill, New York, 1989.

[14] Ashok Dhondu Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization*. Dept. of civil and environmental engineering, University of Iowa, Iowa, Iowa, 1982.

[15] David M. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, New York, 1972.

[16] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms & Engineering Design*. John Wiley & Sons, Inc, New York, 1997.

[17] Carlos A. Coello Coello, Filiberto Santos Hernández, and Francisco Alonso Farrera. Optimal design of reinforced concrete beams using genetic algorithms. *Expert Systems with Applications : An International Journal*, 12(1), January 1997.

[18] Carlos Artemio Coello Coello. *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design*. PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA, apr 1996.

[19] Kalyanmoy Deb. GeneAS: A Robust Optimal Design Technique for Mechanical Component Design. In Dipankar Dasgupta and Zbigniew Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 497–514. Springer-Verlag, Berlin, 1997.

[20] J. F. Fu, R. G. Fenton, and W. L. Cleghorn. A mixed integer-discrete-continuous programming method and its application to engineering design optimization. *Engineering Optimization*, 17(3):263–280, 1991.

[21] Y. J. Cao and Q. H. Wu. Mechanical Design Optimization by Mixed-Variable Evolutionary Programming. In Thomas Bäck, Zbigniew Michalewicz, and Xin Yao, editors, *Proceedings of the 1997 International Conference on Evolutionary Computation*, pages 443–446, Indianapolis, Indiana, 1997. IEEE.

[22] Kalyanmoy Deb. Optimal Design of a Welded Beam via Genetic Algorithms. *AIAA Journal*, 29(11):2013–2015, November 1991.

[23] K. M. Ragsdell and D. T. Phillips. Optimal Design of a Class of Welded Structures Using Geometric Programming. *ASME Journal of Engineering for Industries*, 98(3):1021–1025, 1976. Series B.

[24] James N. Siddall. *Analytical Design-Making in Engineering Design*. Prentice-Hall, 1972.

[25] A. Homaifar, S. H. Y. Lai, and X. Qi. Constrained Optimization via Genetic Algorithms. *Simulation*, 62(4):242–254, 1994.