Engineering Optimization Using a Simple Evolutionary Algorithm

Efrén Mezura-Montes, Carlos A. Coello Coello and Ricardo Landa-Becerra
Evolutionary Computation Group (EVOCINV)
Departamento de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
{emezura,rlanda}@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract

This paper presents a simple $(1 + \lambda)$ Evolution Strategy and three simple selection criteria to solve engineering optimization problems. This approach avoids the use of a penalty function to deal with constraints. Its main advantage is that it does not require the definition of extra parameters, other than those used by the evolution strategy. A self-adaptation mechanism allows the algorithm to maintain diversity during the process in order to reach competitive solutions at a low computational cost. The approach was tested in four well-known engineering design problems and compared against several penalty-function-based approaches and other state-of-the-art technique. The results obtained indicate that the proposed technique is highly competitive in terms of quality, robustness and computational cost.

1. Introduction

Several approaches have been suggested in the literature to solve engineering optimization problems [17]. Evolutionary Algorithms (EAs) are heuristic techniques that have been found particularly useful to solve this kind of optimization problems [9]. However, EAs lack a mechanism able to bias efficiently the search towards the feasible region in constrained search spaces. This has motivated a considerable amount of research and a wide variety of approaches have been suggested in the last few years to incorporate constraints into the fitness function of an evolutionary algorithm [5, 16].

The most common approach adopted to deal with constrained search spaces is the use of penalty functions. When using a penalty function, the amount of constraint violation is used to punish or "penalize" an infeasible solution so that feasible solutions are favored by the selection process. Despite the popularity of penalty functions, they have several drawbacks from which the main one is that they require a careful fine tuning of the penalty factors that accurately estimates the degree of penalization to be applied as to approach efficiently the feasible region [18, 5]. Therefore, other alternative approaches have been suggested. Our approach is based on a simple evolution strategy $(1 + \lambda)$-ES and three simple selection rules to guide the evolutionary search to the feasible region of the search space. Moreover, this approach does not use a penalty function, it does not require the definition of any extra parameters, other than those required by an evolution strategy and it is easy to implement.

This paper is organized as follows: In Section 2 the mathematical definition of the problem is presented. In Section 3, we describe some previous work. A detailed description of our approach is given in Section 4. In Section 5 we describe the selected problems to test our technique. In Section 6, we show the results obtained in the experiments performed and in Section 7 we discuss them. Finally, some conclusions and future work are established in Section 8.

2. Basic Concepts

We are interested in the general non linear programming problem in which we want to: Find $\vec{x}$ which optimizes $f(\vec{x})$ subject to: $g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \;\; h_j(\vec{x}) = 0, \quad j = 1, \ldots, p$ where $\vec{x}$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear). If we de-

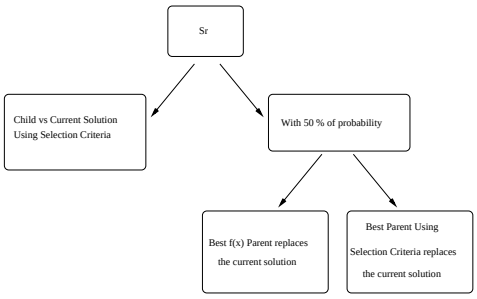

Figure 1. Diagram that illustrates the diversity mechanism implemented for our approach.

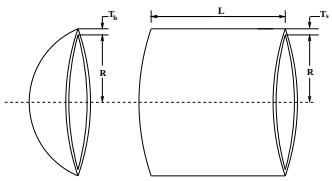

Figure 2. Center and end section of the pressure vessel used for problem 1.

note with $\mathcal{F}$ to the feasible region and with $\mathcal{S}$ to the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$.

3. Previous Work

Several authors have used EAs to solve engineering design problems:

Deb [8] proposed a Genetic Adaptive Search (GeneAS) to solve engineering optimization problems. He proposed to use a both, binary and real encoding for each solution. This approach was tested on three engineering problems [8], making emphasis in problems that have discrete and continuous variables. The obvious drawback of the approach is the need of implementing combined operators for the special encoding adopted.

Coello and Mezura [6] implemented a version of the Niched-Pareto Genetic Algorithm (NPGA) [13] to handle constraints in single-objective optimization problems. The NPGA is a multiobjective optimization approach in which individuals are selected through a tournament based on Pareto dominance. However, unlike the [original] NPGA, Coello and Mezura's approach does not require niches (or fitness sharing [7]) to maintain diversity in the population. The NPGA is a more efficient technique than traditional multiobjective optimization algorithms, because it only uses a sample of the population to estimate Pareto dominance.

Akhtar et al. [1] proposed a swarm-like based approach solve engineering optimization problems. They simulate societies that conform a civilization. In each society there is a leader which is followed by the other members of its society. Besides these societies, there is a leader's society which is formed with the leaders of each society. They are called "general leaders". Constraints are handled by ranking the solutions based on nondominace checking inside their corresponding society. They also use a special operator to allow an individual to be assigned with a variable value that does not exists neither in the leader nor in the solution selected from the society. It can be seen as an individual that is not following its leader. Its main advantage is that the approach requires a low number of evaluations of the objective function to obtain good results, but not the optimum. Its main drawback is that the implementation is not easy. Besides this, the computational cost increases because of the ranking process and the clustering algorithm that the approach requires to initialize the societies.

4. Our approach

Our approach is based on an Evolution Strategy because this technique has been found not only efficient in solving a wide variety of optimization problems [10], but also has a strong theoretical background [3].

The motivation of this work is divided in three parts: (1) We hypothesized that the use of an evolution strategy for constrained optimization would be beneficial to sample the search wide enough, (2) we were aware that having a good mechanism to maintain diversity is one of the keys to produce a constraint-handling approach that is competitive and (3) we did not want to add any extra parameter to the approach in order to make it easy to use.

The three simple selection criteria used in our tournaments are the following (binary tournaments are adopted):

1. Between 2 feasible solutions, the one with the highest fitness value wins.
2. If one solution is feasible and the other one is infeasible, the feasible solution wins.
3. If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

Our approach uses the $1/5$-success rule for self-adapting the σ value of our ES. By using just one σ value and one fitness function evaluation per generation, the resulting computational cost (per generation) is very low.

We use a modified version of a $(1 + \lambda)$-ES [3]. In the original $(1 + \lambda)$-ES, λ solutions are generated from the current solution and if one of the λ is better than the current one, it replaces it. The modifications are the following:

- The selection process was modified in order to allow either infeasible solutions with a good value of the objective function or the best parent (based on the selection criteria) to replace the current solution. Therefore,

besides the λ solutions, they are combined to generate just one child.

- This modified selection process is controlled by a parameter (that is not defined by the user) called Selection Ratio (S_r). This parameter was introduced in [6] and it refers to the percentage of selections that will take place only between the current solution and the child generated by all the λ parents, based on the three selection criteria previously indicated. In the remaining $1 - S_r$ selections, there are two choices: (1) either the parent with the best value of the objective function will replace the current solution (regardless of its feasibility) or (2) the best parent (based on the three selection criteria) will replace the current solution. Both options are given a 50% probability each (see Figure 1).

- The S_r parameter is adapted online using the fitness value of the current solution during an interval of time (number of generations). The "mean deviation" (M_d) of the current solution over a certain number of generations is calculated in order to know how different has been the current solution. All the fitnesses are normalized in order to obtain a value between 0 and 1. The expression to adapt the S_r value is the following:

$$S_r(t) = \begin{cases} S_r(t - \text{interval})/1.001 & \text{if } M_d < 0.1 \\ S_r(t - \text{interval}) * 1.001 & \text{if } M_d > 0.2 \\ S_r(t - \text{interval}) & \text{if } 0.1 \leq M_d \leq 0.2 \end{cases} \quad (1)$$

  where *interval* is defined as a percentage of the maximum number of generations. For example if the interval is defined as 0.05 and the number of generations is 100, the update process will take place at every 5 generations. As can be seen, S_r will be decreased if the current solution has not significantly changed during the given interval (i.e., $M_d < 0.1$) allowing a parent (which may be infeasible) with a good fitness value to replace the current solution. This is meant to increase diversity. On the other hand, S_r will increase if the solution has been significantly different (i.e., $M_d > 0.2$) during the interval, thus favoring deterministic selection to impel convergence. S_r will keep its current value if the variation of the current solution in the interval has been moderated (i.e., $0.1 \leq M_d \leq 0.2$).

- In order to always keep the best solution found during the process a superelitist mechanism is included. Its only goal is to keep the best feasible solution found. This is required because the diversity mechanism adopted makes the current solution to be replaced by another solution which is not necessarily better and may be infeasible. Its implementation does not add any significant extra computational or storage cost to the algorithm.

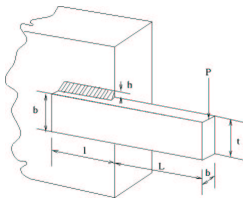

Figure 3. The welded beam used for problem 2.

The pseudo-code of this approach is shown in Figure 4.

Begin
t=0, S_r=0.9, interval=0.9
Create a random initial solution x^0 and store it as the superelitist solution x^S
Evaluate $f(x^0)$
For t=1 to MAX_GENERATIONS **Do**
Produce λ mutations of $x^{(t-1)}$ using:
$x_i^j = x_i^{t-1} + \sigma[t] \cdot N_i(0,1) \; \forall i \in n, j = 1, 2, \ldots, \lambda$
Generate one child x^c by the combination of the λ mutations using
m=randint(1, λ)
$x_i^c = x_i^m, \forall i \in n$
If (flip(S_r)) **Then**
Evaluate f(x^c)
Apply comparison criteria to select the best individual x^t between $x^{(t-1)}$ and x^c
else
Evaluate all the λ mutations obtained
If (flip(0.5)) **Then**
x^t = parent with best objective function value regardless of feasibility
else
x^t = best parent based on the comparison criteria
endif
endif
If (x^t better than x^S) (using selection criteria) **Then**
$x^S = x^t$
endif
$t = t + 1$
If (t mod $n = 0$) **Then**
$\sigma[t] = \begin{cases} \sigma[t-n]/c & if\ p_s > 1/5 \\ \sigma[t-n] \cdot c & if\ p_s < 1/5 \\ \sigma[t-n] & if\ p_s = 1/5 \end{cases}$
End If
If (t mod interval $= 0$) **Then**
Calculate the mean deviation (M_d) of the current solutions in the interval
If ($M_d < 0.1$) **Then**
$S_r(t) = S_r(t - \text{interval})/1.001$
else
If ($M_d > 0.2$) **Then**
$S_r(t) = S_r(t - \text{interval})/1.001$
End If
End If
End If
End For
End

Figure 4. SES algorithm (n is the number of decision variables of the problem, $flip(P)$ is a function that returns TRUE with probability P).

5. Test Problems

To test our technique we decided to implement four penalty-based approaches: Death penalty, a static penalty [12], a dynamic penalty [14] and an adaptive penalty [11].

We selected four well known engineering design problems to validate our approach . The full description of each of them is provided below:

- **Problem 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 [15], the problem can be stated as follows:

  Minimize : $f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

  Subject to :

  $g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0$

  $g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0$

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

  $g_4(\vec{x}) = x_4 - 240 \leq 0$

  where $1 \leq x_1 \leq 99$, $1 \leq x_2 \leq 99$, $10 \leq x_3 \leq 200$ y $10 \leq x_4 \leq 200$.

- **Problem 2: (Design of a Welded Beam)**

  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 [17]. There are four design variables as shown in Figure 3 [17]: h (x_1), l (x_2), t (x_3) and b (x_4).

  The problem can be stated as follows:

  Minimize: $f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

  Subject to:

  $g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leq 0$

  $g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leq 0$

  $g_3(\vec{x}) = x_1 - x_4 \leq 0$

  $g_4(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$

  $g_5(\vec{x}) = 0.125 - x_1 \leq 0$

  $g_6(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leq 0$

  $g_7(\vec{x}) = P - P_c(\vec{x}) \leq 0$

  where $\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$

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

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

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

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

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

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

  where $0.1 \leq x_1 \leq 2.0$, $0.1 \leq x_2 \leq 10.0$, $0.1 \leq x_3 \leq 10.0$ y $0.1 \leq x_4 \leq 2.0$.

- **Problem 3: (Minimization of the Weight of a Tension/Compression String)**

  This problem was described by Arora [2] and Belegundu [4], and it consists of minimizing the weight of a tension/compression spring (see Figure 5) 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 (x_2), the wire diameter d (x_1) and the number of active coils N (x_3). Formally, the problem can be expressed as:

  Minimize: $(N + 2)Dd^2$

  Subject to:

  $g_1(\vec{x}) = 1 - \frac{D^3N}{71785d^4} \leq 0$

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

  $g_3(\vec{x}) = 1 - \frac{140.45d}{D^2N} \leq 0$

  $g_4(\vec{x}) = \frac{D + d}{1.5} - 1 \leq 0$

  where $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$ y $2 \leq x_3 \leq 15$.

- **Problem 4: (Minimization of the Weight of a Speed Reducer)**

  The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surfaces stress, transverse deflections of the shafts and stresses in the shafts. The variables $x_1, x_2, \cdots, x_7$ are the face width, module of teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings and the diameter of the first and second shafts. The third variable is integer, the rest of them are continuous.

  Minimize : $f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$

  Subject to :

  $g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$

  $g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$

  $g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$

  $g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$

  $g_5(\vec{x}) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right)^{1/2}}{110.0x_6^3} - 1 \leq 0r$

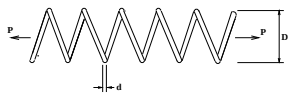

Figure 5. Tension/compression string used for problem 3

$$g_6(\vec{x}) = \frac{\left(\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5\times10^6\right)^{1/2}}{85.0x_7^3} - 1 \leq 0$$
$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \leq 0$$
$$g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \leq 0$$
$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \leq 0$$
$$g_{10}(\vec{x}) = \frac{1.5x_6+1.9}{x_4} - 1 \leq 0$$
$$g_{11}(\vec{x}) = \frac{1.1x_7+1.9}{x_5} - 1 \leq 0$$

where $2.6 \leq x_1 \leq 3.6, 0.7 \leq x_2 \leq 0.8, 17 \leq x_3 \leq 28, 7.3 \leq x_4 \leq 8.3, 7.8 \leq x_5 \leq 8.3, 2.9 \leq x_6 \leq 3.9$ and $5.0 \leq x_7 \leq 5.5$.

6. Comparison of Results

A total of 30 runs per technique per problem were performed. The number of evaluations of the objective function was fixed to $36,000$ for the four penalty-based approaches and also for our approach. For the penalty-based approaches we used a gray-coded genetic algorithm with roulette wheel selection, one point crossover and uniform mutation. The population size was 100 individuals and the number of generations 360. The rate of crossover was 0.6 and the mutation rate was 0.01. The parameters for the dynamic and adaptive approaches were defined after a trial-and-error process. The reported parameters were those which provided better results and they are the following: Dynamic approach: $\alpha = 2$, $\beta = 2$, $C = 0.5$. Adaptive approach: $\beta_1 = 2.0$, $\beta_2 = 4.0$, $k = 50$, $\delta_{initial} = 5000$

The initial values for the $(1+\lambda)$-ES parameters were: initial stepsize value $\sigma = 4.0$, the factor of update of stepsize $C = 0.99$, number of parents generated $\lambda = 3$, and maximum number of generations = $30,000$. The interval of the S_r updates was 0.1. It means that the update will take place at every 3000 generations (10% of the total number of generations). In every run, the initial value for the S_r is 0.9. The results obtained are shown in Table 1 for the Pressure Vessel Design, in Table 2 for the Welded Beam design, in Table 3 for the Tension/Compression Spring design and, finally, in Table 4 for the Speed Reducer Design Problem.

7. Discussion of Results

The discussion is divided in two parts: (1) comparison of our simple evolution strategy against the penalty-based approaches and (2) comparison of our approach against the Socio-Behavioral approach [1]. Both comparisons are based on two aspects: quality and robustness of the solutions.

In the Pressure Vessel problem, our evolution strategy found the best result and it was also the most robust (Table 1). The dynamic penalty approach ranked second. In third place was the death penalty approach. The remaining techniques failed to reach the feasible region consistently (Table 5). For the Welded Beam problem, our technique was slightly surpassed by the static approach. Nonetheless, our approach was the most robust (Table 2).

For the Spring problem, our approach found the best solution and it was the second most consistent approach, only slightly surpassed by the adaptive penalty approach. For the last problem, the four penalty-based approaches failed to reach the feasible region of the problem. However, the Simple Evolution Strategy found it in every single run (Table 5).

These results evidence the lack of consistency of the penalty-function-based approaches (based on the difficult to define their required parameters). On the other hand, the

Problem 1	Details of the best solution found	
	Our approach	SB
x_1	0.812500	0.8125
x_2	0.437500	0.4375
x_3	42.098370	41.9768
x_4	176.637146	182.2845
$g_1(x)$	-0.000001	-0.0023
$g_2(x)$	-0.035882	-0.0370
$g_3(x)$	-0.835772	-23420.5966
$g_4(x)$	-63.362858	-57.7155
$f(x)$	**6059.714355**	**6171.0**

Table 7. Best solution found for the Pressure Vessel Design Problem.

Problem 2	Details of the best solution found	
	Our approach	SB
x_1	0.203642	0.2407
x_2	3.593228	6.4851
x_3	8.980190	8.2399
x_4	0.208393	0.2497
$g_1(x)$	-165.394302	-129.8545
$g_2(x)$	-10.008150	-270.4023
$g_3(x)$	-0.004751	$-.009008$
$g_4(x)$	-3.411678	-2.9663
$g_5(x)$	-0.078642	-0.1157
$g_6(x)$	-0.235454	-0.2343
$g_7(x)$	-210.369049	-372.4990
$f(x)$	**1.748594**	**2.4426**

Table 8. Best solution found for the Welded Beam Design Problem.

Pressure Vessel Design	Statistical Comparison				
	Death Penalty	Static	Dynamic	Adaptive	Our approach
Best	6129.827637	52.177204*	6104.700195	88.063927*	6059.714355
Mean	7191.641992	53.757520	6670.045085	3335.592351	6355.343115
Median	7239.383545	52.870346*	6688.087646	604.228973*	6392.557129
Worst	8876.304688	67.266701*	7788.871094	11983.214844	6846.628418
St. Dev	636.043280	2.897835	397.492272	4172.602199	256.043795

Table 1. Statistical values obtained with each approach for the Pressure Vessel design problem. "*" means infeasible

Welded Beam Design	Statistical Comparison				
	Death Penalty	Static	Dynamic	Adaptive	Our approach
Best	1.748899	1.742388	1.752588	1.657890*	1.748594
Mean	2.032251	1.910210	2.067771	2.108092	1.870860
Median	1.998613	1.876692	1.982786	2.016899	1.852371
Worst	2.973882	2.252468	2.855598	3.351592*	2.232832
St. Dev	0.263312	0.135383	0.277771	0.361124	0.106377

Table 2. Statistical values obtained with each approach for the Welded Beam design problem. "*" means infeasible

Ten./Comp. Spring Design	Statistical Comparison				
	Death Penalty	Static	Dynamic	Adaptive	Our approach
Best	0.012732	0.012729	0.012689	0.012729	0.012688
Mean	0.014527	0.013774	0.013681	0.013675	0.013014
Median	0.014141	0.013621	0.013436	0.013606	0.012756
Worst	0.017723	0.016407	0.016597	0.015933	0.017037
St. Dev	0.001457	0.001045	0.000934	0.000720	0.000801

Table 3. Statistical values obtained with each approach for the Tension/Compression Spring design problem.

Speed Reducer Design	Statistical Comparison				
	Death Penalty	Static	Dynamic	Adaptive	Our approach
Best	1399.237427*	1671.386475*	1265.959229*	1679.101318*	3025.005127
Mean	1399.237427	267997861.844320	20032.261825	12174.159896	3088.777816
Median	524036.802128*	3661.611328*	3179.876343*	3272.115234	3078.591797
Worst	2913.118042*	8024214016.0*	243416.062500*	244393.671875*	3226.248291
St. Dev	15600199.00	1440295898.14	59873.421388	43472.043832	47.361890

Table 4. Statistical values obtained with each approach for the Speed Reducer design problem. "*" means infeasible

Problem	Percentage of runs in wich the feasible region was reached				
	Death Penalty	Static	Dynamic	Adaptive	Our approach
Pressure Vessel	100%	0%	100%	33%	100%
Welded Beam	100%	100%	100%	80%	100%
Spring	100%	100%	100%	100%	100%
Speed Reducer	0%	0%	0%	0%	100%

Table 5. Percentage of runs that converged to a feasible solution.

	Best Result		Mean Result		Worst Result	
Problem	Ours	SB	Ours	SB	Ours	SB
Press.Vessel	6059.728027	6171.00	6295.362964	6335.05	6887.999023	6453.65
Welded Beam	1.742510	2.4426	1.876213	2.5215	2.023696	2.6315
Speed Red.	3038.684082	3008.08	3088.530505	3012.12	3199.294922	3028.28

Table 6. Comparison of Results Between Our Approach and the Socio Behavioral approach [1]

Problem 4	Details of the best solution found	
	Our approach	SB
x_1	3.506163	3.503122
x_2	0.700831	0.700006
x_3	17	17
x_4	7.460181	7.549126
x_5	7.962143	7.859330
x_6	3.362900	3.365576
x_7	5.308949	5.289773
$g_1(x)$	-0.077734	-0.075548
$g_2(x)$	-0.201305	-0.199413
$g_3(x)$	-0.474119	-0.456175
$g_4(x)$	-0.897068	-0.899442
$g_5(x)$	-0.011021	-0.013213
$g_6(x)$	-0.012500	-0.001740
$g_7(x)$	-0.702147	-0.702497
$g_8(x)$	-0.000573	-0.001738
$g_9(x)$	-0.583095	-0.582608
$g_{10}(x)$	-0.069144	-0.079580
$g_{11}(x)$	-0.027920	-0.017887
$f(x)$	**3025.005127**	**3008.08**

Table 9. Best solution found for the Speed Reducer Design Problem.

Simple Evolution Strategy obtained the lowest standard deviations so far. Also, the quality of results of our approach, generally speaking, was also better.

The comparison against the Socio-Behavioral Approach (SB) was made using only three of the four problems, because there were no results available for the Spring problem. As can be seen in Table 6, our technique provided better quality and robustness of results on two of three problems. For the Speed Reducer problem, our approach was close to the best solution found by the SB. However, despite the fact that the SB approach performs less evaluations of the objective function ($20,000$), the additional computational cost derived from the ranking and clustering process

Statistical Results	with $\lambda = 3$ and No. of generations=$60,000$			
	Pressure V.	Beam	Spring	Speed Red.
Best	6059.898926	1.739163	0.012680	2998.011963
Mean	6238.507764	1.885300	0.013050	3056.206999
Median	6187.255615	1.860463	0.012738	3066.821777
Worst	6556.407227	2.183147	0.015465	3162.881104
St. Dev	158.320180	0.112471	0.000672	49.406654

Table 10. Statistical values obtained for the four design problem when the number of generations is increased to $60,000$ with 30 independent runs.

makes the computational cost of our approach to be lower than the SB technique. Also, we argue that our algorithm is much easier to implement.

In order to know if both the quality and robustness of the approach get better when the number of evaluations increases, we performed 2 experiments: (1) Increase the number of generations to the double ($60,000$) while fixing the value of $\lambda = 3$ and (2) increase the value of $\lambda = 6$ while the number of generations remained unchanged ($30,000$). The results are shown in Tables 10 and 11. It is clear that there is a moderate improvement in the robustness of the results when the number of generations increases and there is a improvement in the quality of the results in problem 4. On the other hand, when we increase the number of parents in the population, there is just a slight improvement in the robustness and the quality of the results. Then, we can say that, in general, by increasing the number of generations, our approach improves in terms of robustness and slightly in terms of quality.

8. Conclusions and Future Work

A novel approach to solve engineering design problems based on a simple evolution strategy was presented. The

Statistical Results	with $\lambda = 6$ and No. of generations=$30,000$			
	Pressure V.	Beam	Spring	Speed Red.
Best	6060.504883	1.735141	0.012692	3013.404297
Mean	6242.881201	1.844146	0.013468	3072.620256
Median	6140.236328	1.818958	0.013058	3059.338379
Worst	7052.284668	2.046225	0.016100	3163.569824
St. Dev	213.559758	0.085546	0.000983	36.736896

Table 11. Statistical values obtained for the four design problem when $\lambda = 6$ with 30 independent runs.

main advantage of our approach is that it does not require a penalty function or any extra parameters (other than the original parameters of an evolution strategy). Also, the computational cost of our approach (measured in terms of the number of evaluations of the objective function) is very low ($36,000$). Furthermore, the proposed approach is very simple and easy to implement. Our simple evolution strategy provided better results than traditional penalty-based approaches and it was very competitive with respect to an algorithm representative of the state-of-the-art in evolutionary optimization.

Our future paths of research are to test the approach in other real world problems (with a high dimensionality) and to improve its local search power in order to obtain results of even higher quality at a lower computational cost.

Acknowledgments

The first and third authors acknowledge support from the mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through a scholarship to pursue graduate studies at CINVESTAV-IPN's Electrical Engineering Department. The second author acknowledges support from CONACyT through project number 32999-A.

References

[1] S. Akhtar, K. Tai, and T. Ray. A Socio-Behavioural Simulation Model for Engineering Design Optimization. *Engineering Optimization*, 34(4):341–354, 2002.

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

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[4] A. D. Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization*. Department of civil and environmental engineering, University of Iowa, Iowa, Iowa, 1982.

[5] C. A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[6] C. A. Coello Coello and E. Mezura-Montes. Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments. In I. Parmee, editor, *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)*, volume 5, pages 273–284, University of Exeter, Devon, UK, April 2002. Springer-Verlag.

[7] K. Deb and D. E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.

[8] K. Deb and M. Goyal. A Combined Genetic Adaptive Search GeneAS for Engineering Design. *Computer Science and Informatics*, 26(4):30–45, 1996.

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

[10] M. Gorges-Schleuter, I. Sieber, and W. Jakob. Local interaction evolution strategies for design optimization. In *1999 Congress on Evolutionary Computation*, pages 2167–2174, Piscataway, NJ, 1999. IEEE Service Center.

[11] A. B. Hadj-Alouane and J. C. Bean. A Genetic Algorithm for the Multiple-Choice Integer Program. *Operations Research*, 45:92–101, 1997.

[12] F. Hoffmeister and J. Sprave. Problem-independent handling of constraints by use of metric penalty functions. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96)*, pages 289–294, San Diego, California, February 1996. The MIT Press.

[13] J. Horn, N. Nafpliotis, and D. E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[14] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In D. Fogel, editor, *Proceedings of the first IEEE Conference on Evolutionary Computation*, pages 579–584, Orlando, Florida, 1994. IEEE Press.

[15] 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.

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

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

[18] A. E. Smith and D. W. Coit. Constraint Handling Techniques—Penalty Functions. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.