Useful Infeasible Solutions in Engineering Optimization with Evolutionary Algorithms

Efrén Mezura-Montes and Carlos A. Coello Coello

CINVESTAV-IPN, Evolutionary Computation Group (EVOCINV),
Computer Science Section, Electrical Engineering Department,
Av. IPN No. 2508 Col, San Pedro Zacatenco México D.F. 07300, México
emezura@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract. We propose an evolutionary-based approach to solve engineering design problems without using penalty functions. The aim is to identify and maintain infeasible solutions close to the feasible region located in promising areas. In this way, using the genetic operators, more solutions will be generated inside the feasible region and also near its boundaries. As a result, the feasible region will be sampled well-enough as to reach better feasible solutions. The proposed approach, which is simple to implement, is tested with respect to typical penalty function techniques as well as against state-of-the-art approaches using four mechanical design problems. The results obtained are discussed and some conclusions are provided.

1 Introduction

Evolutionary Algorithms (EAs) have been widely used to solve optimization problems [1]. We are interested in the general non linear programming problem in which we want to:Find $\boldsymbol{x}$ which optimizes $f(\boldsymbol{x})$ subject to: $g_i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, m\ h_j(\boldsymbol{x}) = 0,\ j = 1, \ldots, p$ where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of solutions $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$, where each $x_i,\ i = 1, \ldots, n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$; m 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 denote with $\mathcal{F}$ to the feasible region and with $\mathcal{S}$ to the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$. It is well-known that, in their original versions, EAs lack a mechanism to deal with constrained search spaces. Hence, a considerable amount of research have been focused on incorporating the constraints of the problem into the fitness function of an EA. The most common approach is the use of a penalty function [2], whose aim is to decrease the fitness of infeasible solutions in order to favor the selection of the feasible ones (the fitness of a solution is calculated by merging the objective function value and the sum of constraint violation multiplied by a penalty factor). However, its main drawback is the careful fine-tuning required by the penalty factors, whose values indicate the degree of penalization [2]. In this paper, we propose to avoid the use of penalty functions; instead, we propose to handle the objective function value and the constraints of the problem separately and to use a mechanism to keep a few infeasible solutions with the lowest sum of constraint violation in the population for the next generation. These infeasible solutions must have

A. Gelbukh, A. de Albornoz, and H. Terashima (Eds.): MICAI 2005, LNAI 3789, pp. 652–662, 2005.

the best objective function value among infeasible solutions. The aim is to promote the generation of solutions close to the boundaries of the feasible region in order to reach a better solution, despite its location inside or in the boundaries of the feasible set. The paper is organized as follows: Section 2 summarizes some approaches found in the literature of engineering design with evolutionary algorithms. In Section 3 we present details of the proposed approach, including a simple example to show its behavior. The experimental design, the results obtained and a discussion about them are provided in Section 4. Finally, in Section 5 we present our conclusions and future work.

2 Related Work

Many successful applications of EAs to solve engineering design problems have been reported. Ray & Liew [3] used a swarm-like based approach to solve engineering optimization problems. In their approach, a civilization was conformed by several societies, whose leaders guide the members of each society. Besides, there is a leaders' society which contains all the leaders of each society. Constraints are handled using a dominance-based approach in the constraints space [3]. To maintain diversity, the authors propose a mechanism to allow an individual not to follow its leader. The main advantage of the approach is that it requires a low number of evaluations of the objective function to obtain competitive results. However, the computational cost is increased by the ranking process and the clustering algorithm that the approach requires to initialize the societies. Hernández et al. [4] proposed PASSSS, an approach based on a multi-objective optimization algorithm called PAES [4] to solve some benchmark problems and also some structural design problems. The approach uses an external memory to store the best set of solutions found. Furthermore, PASSSS requires a shrinking mechanism to reduce the search space. Pareto dominance is used only to decide whether or not a new solution is inserted in the external memory. The authors acknowledge that the most important mechanisms of IS-PAES are its shrinking procedure and the information provided by the external memory which is used to decide the shrinking of the search space. Furthermore, despite its good performance as a global optimizer, PASSSS is an approach far from being simple to implement. An improved particle swarm optimization was proposed by He et al. [5] to solve mechanical design optimization problems. The idea is to let the particles fly only inside the feasible region. Therefore, an initial feasible population is required, which is the main disadvantage of the approach, because for some problems, even generating one single feasible solution is very difficult. Its main advantage is that its computational cost is relatively low compared with the approaches mentioned before.

3 Our Approach

The main motivation of this work is to avoid the use of a penalty function to handle the constraints of the problems and to provide a simple mechanism capable of boosting the generation of solutions close to the boundaries of the feasible region of the search space. Our proposed approach works in the following way: At each generation, the solutions are ranked based on three criteria:

1. Between 2 feasible solutions, the one with the highest fitness value wins (assuming a maximization problem/task).
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 ($\sum_{i=1}^{m} max(0, g_i(\boldsymbol{x}))$).

After the ranking process, the selected individuals for the next generation will be (1) those feasible solutions with a more promising value of the objective function and (2) infeasible solutions with the lowest value of the sum of constraint violation. As the population evolves, this selection process will lead the search to reach faster the feasible region (like a severe penalty function). However, in order to maintain infeasible solutions close to the the feasible region, at each generation, the infeasible solution with the lowest sum of constraint violation and with the best value of the objective function (taken from the μ parents or the λ offspring, each one with 50% probability) will be included in the population for the next generation. This mechanism is controlled by a user-defined parameter. Therefore, more than one copy of the same infeasible solution will be in the same population. However, it is a desired behavior because this solution will have more probabilities to generate more individuals close to it and this promising area will be explored more in-depth. As a result, the population will have, most of the time, a few infeasible solutions located in promising areas of the boundaries of the feasible region. If all the population is feasible, a random solution will be copied in the population. The parameter which controls this mechanism was called by us as Selection ratio: ($0 \leq S_r \leq 1$). We used a $(\mu + \lambda)$ evolution strategy as a search engine, because its selection mechanism fits with our proposal (the selection is made to create the population for the next generation). We used typical genetic operators: self-adaptive Gaussian mutation [6] and we combined panmictic discrete/intermediate recombination for the decision variables and control variables (encoded mutation values) as well [6]. A pseudocode of the approach is provided in Figure 1.

```
   Begin
      Create a random initial population of μ solutions
      For G=1 to MAX_GENERATIONS Do
         For i=1 to |λ| Do
               Choose randomly one parent from the μ available plus n other parents (one per variable)
               Generate one offspring by panmictic discrete/intermediate recombination
               Mutate the offspring
         End For
         Sort the (μ + λ) solutions using the three criteria based on feasibility
         For i=1 to |μ| Do
⇒           If flip(S_r) Then
               Move the solution at the top of the sorted (μ + λ) to the population for the next generation
            Else
⇒              Copy the best infeasible solution (from parents or offspring) to the population for the next generation
            End If
         End For
         G = G + 1
      End For
   End
```

Fig. 1. Our approach. flip(W) returns 1 with probability W. Arrows indicate the steps added to maintain infeasible solutions.

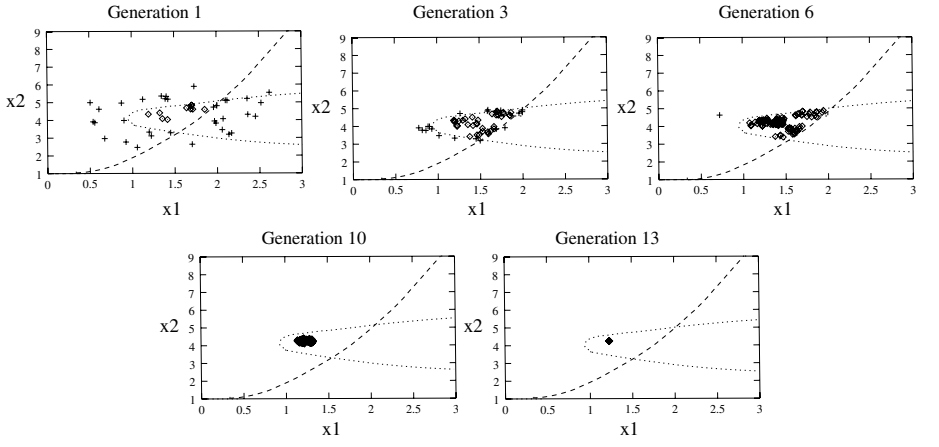

Fig. 2. Graphs showing the population behavior using our proposed mechanism. "◇" points are feasible solutions, "+" points are infeasible ones. The dashed line represents constraint $g_1(x)$ of the problem and the dotted line represents constraint $g_2(x)$.

A graphical example of the expected behavior of the approach can be found in Figure 2. We used a 2-dimensional test problem, which is a problem easy to solve by the approach; it requires about 5400 evaluations of the objective function to reach the global optimum, but it helps to visualize how our approach works. The definition of this problem is the following:

Maximize: $f(\boldsymbol{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$
subject to: $g_1(\boldsymbol{x}) = x_1^2 - x_2 + 1 \leq 0$ $g_2(\boldsymbol{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$

where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The global optimum is located at $x^* = (1.2279713, 4.2453733)$ where $f(x^*) = 0.095825$. As it can be observed, in generation 1 there are a few feasible as well as several infeasible solutions. The behavior of the approach can be observed in generation 3, where there are more feasible solutions than those in generation 1 and also there are infeasible solutions surrounding the feasible region. In this way, the feasible region is sampled well-enough as to find promising areas (three areas in the example). This is shown in generation 6, where there is still an infeasible solution in the population. It is worth noticing that this infeasible solution is close to the area where the global optimum is located; this can be seen in generation 10 where the infeasible solution has disappeared but the approach has found the vicinity of the constrained global optimum. Our algorithm has converged to the constrained global optimum in generation 18.

4 Experiments and Discussion

Our experimental design has two parts: (1) to compare our approach against different types of penalty function approaches and (2) to compare our results against state-of-the-art approaches. We selected four well known engineering design problems to use them

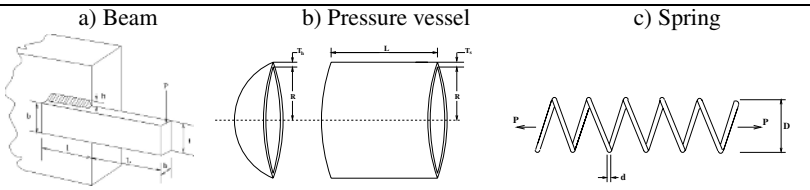

Fig. 3. Figures of the design problems adopted

in the experiments. The full description of each of them is provided in the appendix at the end of this paper.

In the first part of our experiments, we decided to implement four penalty-based approaches: Death penalty (to assign a zero fitness to infeasible solutions) [7], a static penalty (fixed penalty factor during all the process) [8], a dynamic penalty (the penalty factor is initialized with a low value and it is increased as the process evolves) [9] and an adaptive penalty (the penalty factor is adapted according to the number of feasible solutions in the population) [10]. A total of 30 runs per technique per problem were performed. The number of evaluations of the objective function was fixed to $30,000$ 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 static, dynamic and adaptive approaches were defined after a trial-and-error process. The reported parameters were those which provided the best results and they are the following: Static approach: fixed penalty factor $= 1000$. Dynamic approach: $\alpha = 2$, $\beta = 2$, $C = 0.5$. Adaptive approach: $\beta_1 = 2.0$, $\beta_2 = 4.0$, $k = 50$, $\delta_{initial} = 5000$ For our approach we used a $(15 + 100)$-ES with the following initial parameters: generations = 300, $S_r = 0.97$ (which means that 3 times every 100 selections, the best infeasible solution will be copied into the population for the next generation). This small value was chosen based on conclusions found in the literature which show that only a few infeasible solutions are enough to improve performance [11]. The learning rates values were calculated using [6] (where n is the number of decision variables of the problem): $\tau = (\sqrt{2\sqrt{n}})^{-1}$ and $\tau' = (\sqrt{2n})^{-1}$. In order to favor finer movements in the search space, we initialized the mutation stepsizes for all the individuals in the initial population with only a 40% of the value obtained by the following formula (where n is the number of decision variables): $\sigma_i(0) = 0.4 \times (\Delta x_i/\sqrt{n})$ where Δx_i is approximated as follows, $\Delta x_i \approx x_i^u - x_i^l$, where $x_i^u - x_i^l$ are the upper and lower limits of the decision variable i. Discrete variables were handled by just truncating the real value to its closest integer value. The statistical results of the 30 independent runs are shown in Table 1.

As it can be seen, our approach was the only to find feasible solutions in all runs for the four problems. No penalty function was able to find feasible solutions for problem 4, the static approach could not find feasible solutions for problem 2, the dynamic approach failed to reach the feasible region in two runs for problem 1 and the adaptive approach found feasible solutions in only 27 runs for problems 1 and 2. The results in Table 1 also reflect how our approach outperforms the four penalty-based approaches in

Table 1. Statistical values obtained with each approach for the four design problem. "-" means no feasible solutions found. A result in **boldface** means a better result. "*" and "**"mean that only in 28 and 27 runs (out of 30) feasible solutions were found, respectively.

a) Welded beam design

	Death Penalty	Static	Dynamic *	Adaptive **	Our approach
Best	1.747810	1.734624	1.793387	1.776909	**1.724852**
Mean	2.021429	1.925931	2.182719	2.043387	**1.777692**
Worst	2.710572	2.320612	3.731899	3.807526	**2.074562**
St. Dev	2.3E-1	1.6E-1	4.5E-1	4.1E-1	**8.8E-2**

b) Pressure vessel design

	Death Penalty	Static	Dynamic	Adaptive **	Our approach
Best	6171.813965	–	6162.862793	6242.527124	**6059.701610**
Mean	7429.709001	–	7042.828564	7241.953700	**6379.938037**
Worst	9763.333984	–	7798.198242	12142.707901	**6820.397461**
St. Dev	7.9E+2	–	5.3E+2	1.2E+3	**2.1E+2**

c) Ten./Comp. spring design

	Death Penalty	Static	Dynamic	Adaptive	Our approach
Best	0.012727	0.012716	0.012690	**0.012684**	0.012689
Mean	0.014870	0.014319	0.013589	0.013406	**0.013165**
Worst	0.018671	0.017603	0.016827	0.015420	**0.014078**
St. Dev	1.7E-3	1.4E-3	1.0E-3	7.5E-4	**3.9E-4**

d) Speed reducer design

	Death Penalty	Static	Dynamic	Adaptive	Our approach
Best	–	–	–	–	**2996.348094**
Mean	–	–	–	–	**2996.348094**
Worst	–	–	–	–	**2996.348094**
St. Dev	–	–	–	–	**0**

quality (best solution found) and robustness ("best" mean, worst and standard deviation values) in all problems. The only exception was problem 3, where the adaptive penalty approach found a slightly "better" best solution. In order to analyze the convergence behavior, in Figure 4 we show the convergence graph obtained by each penalty approach and by our approach as well, for the first three test problems (the speed reducer is omitted because none penalty approach reached the feasible region). Our approach (line with black points) seems to converge really fast to a promising area (before generation 50) and the remainder of the process, this solution is slightly improved (this is truth for the three graphs). On the other hand, for problem 1 (welded beam) and problem 2 (pressure vessel), the penalty approaches got trapped in local optima or had either a very irregular convergence or did not converge at all. Finally, in problem 3 (spring) the compared techniques had problems to reach promising areas at the beginning, and at the end, they could not reach equally good solutions as those found by our algorithm.

The overall results suggest that the proposed approach was able to provide a consistent performance, while the penalty-based approaches sometimes were competitive but in other cases, their results were poor. This is due to the fact that the penalty factors must be updated according to the problem to be solved, and our approach seems to be more stable using the same set of parameters for all test problems.

The statistical results of the second part of our experiments are summarized in Table 2, where we compared our obtained solutions against those provided by the Particle Swarm Optimizer of He et al. [5] and with respect to the Civilization simulation

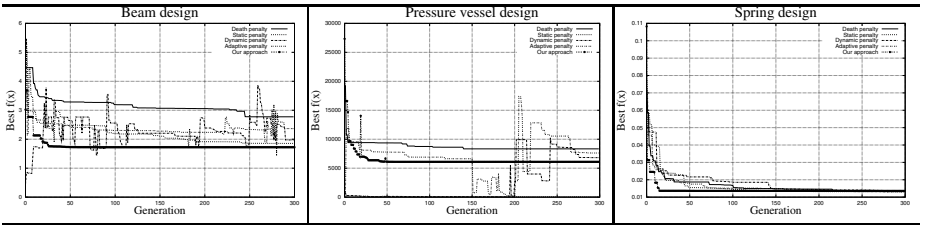

Fig. 4. Convergence graphs for three problems using the four penalty functions and our approach

Table 2. Comparison of results with respect to two state-of-the-art approaches. A result in **boldface** means a better result.

Problem	Stats	Ray & Liew [3]	He et al. [5]	Our approach
Welded beam	best	2.385435	2.380957	**1.724852**
	mean	3.255137	2.381932	**1.777692**
	St. Dev	9.6E-1	**5.2E-3**	8.8E-2
	evaluations	33000	**30000**	**30000**
Pressure vessel	best	6171.00	6059.7143	**6059.701610**
	mean	6335.05	**6289.92881**	6379.938037
	St. Dev	NA	3.1E+2	**2.1E+2**
	evaluations	**20000**	30000	30000
Ten/Comp. spring	best	0.012669	**0.012665**	0.012689
	mean	0.012923	**0.012702**	0.013165
	St. Dev	5.9E-4	**4.1E-5**	3.9E-4
	evaluations	25167	**15000**	30000
Speed reducer	best	**2994.744241**	NA	2996.348094
	mean	3001.758264	NA	**2996.348094**
	St. Dev	4.0E+0	NA	**0**
	evaluations	54456	NA	**30000**

of Ray & Liew [3]. The number of evaluations required by each approach is included. Also, in Table 3 we provide the details of the best solution found by each technique.

Based on the information in Table 2, our approach provided the best performance in problem 1 (beam design), "better" best, and mean result using a lower number of evaluations. For problem 2 (vessel design), none of the approaches was a clear winner. Ray's approach used the lowest number of evaluations of the objective function, the "best" mean value was provided by He's approach and our approach obtained the "best" best solution and the lowest standard deviation value. For problem 3 (spring design), He's approach was the most competitive. Finally, for problem 4 (speed reducer design), the best result was found by Ray's technique, but the most robust performance ("better" mean, worst and standard deviation value) and the lowest number of evaluations required was provided by our approach. He's results were not available for this problem.

From the details of the best solution found by each approach (Table 3) we emphasize the following: In the beam design problem, our approach was able to find a better result, which is located in the boundaries of the feasible region (see the values close to zero for constraints 1 and 7) and the compared approaches could not do that. For the vessel and the spring design problems, the results obtained by the compared algorithms are very similar. In the first case (vessel) our approach was able to provide the

Table 3. Details of the best solution found by each compared state-of-the-art technique

Welded beam	Problem 1		
	Ray & Liew [3]	He et al. [5]	Our approach
x_1	0.244438	0.244369	0.205730
x_2	6.237967	6.217520	3.470489
x_3	8.288576	8.291471	9.036624
x_4	0.244566	0.244369	0.205729
$g_1(x)$	−5760.110471	−5741.176933	0.000000
$g_2(x)$	−3.245428	0.000001	0.000002
$g_3(x)$	−0.000128	0.000000	0.000000
$g_4(x)$	−3.020055	−3.022955	−3.432984
$g_5(x)$	−0.119438	−0.119369	−0.080730
$g_6(x)$	−0.234237	−0.234241	−0.235540
$g_7(x)$	−13.079305	−0.000309	0.000001
f(x)	**2.38119**	**2.380956**	**1.724852**

Pressure vessel	Problem 2		
	Ray & Liew [3]	He et al. [5]	Our approach
x_1	0.8125	0.8125	0.8125
x_2	0.4375	0.4375	0.4375
x_3	41.9768	42.098446	42.098446
x_4	182.2845	176.636052	176.636596
$g_1(x)$	−0.0023	−0.000000	0.000000
$g_2(x)$	−0.0370	−0.035881	−0.035880
$g_3(x)$	−23420.5966	−0.000000	0.000000
$g_4(x)$	−57.7155	−63.363948	−63.363404
f(x)	**6171.0**	**6059.701610**	**6059.7143**

Ten./Comp. spring	Problem 3		
	Ray & Liew [3]	He et al. [5]	Our approach
x_1	0.0521602	0.051690	0.052836
x_2	0.368159	0.356750	0.384942
x_3	10.648442	11.287126	9.807729
$g_1(x)$	−0.000000	−0.000000	−0.000001
$g_2(x)$	−0.000000	0.000000	−0.000000
$g_3(x)$	−4.075805	−4.053827	−4.106146
$g_4(x)$	−0.719787	−0.727706	−0.708148
f(x)	**0.012669**	**0.012665**	**0.012689**

Speed reducer	Problem 4	
	Ray & Liew [3]	Our approach
x_1	3.500000	3.499999
x_2	0.700000	0.699999
x_3	17	17
x_4	7.327602	7.300000
x_5	7.715321	7.800000
x_6	3.350267	3.350215
x_7	5.286655	5.286683
$g_1(x)$	NA	−0.073915
$g_2(x)$	NA	−0.197998
$g_3(x)$	NA	−0.499172
$g_4(x)$	NA	−0.901472
$g_5(x)$	NA	−0.000000
$g_6(x)$	NA	−0.000000
$g_7(x)$	NA	−0.702500
$g_8(x)$	NA	0.000000
$g_9(x)$	NA	−0.583333
$g_{10}(x)$	NA	−0.051325
$g_{11}(x)$	NA	−0.010852
f(x)	**2994.744241**	**2996.348094**

best of them. However, for the second case (spring), our approach provided the worst of them. We argue that the approach requires (at least for this problem) more infeasible solutions in the population. This issue is part of our future work. For the last problem, our approach was able to explore the boundaries of the feasible region, but again, it did not find a better result than that found by Ray's technique. It is important to highlight that He's approach is designed to move only inside the feasible region of a given problem. Therefore, it requires a feasible initial population (which, for some problems could be very difficult to get). Ray's approach adds extra computational cost derived of clustering routines. On the other hand, our approach is based on a simple modification (to maintain the lowest infeasible solution in the population) to an EA and, therefore it is easy to implement. Besides, it does not add considerable extra computational cost. Furthermore, our approach does not require to have feasible solutions at the beginning.

We can conclude for this second part of the experiments that our approach is able to explore the boundaries of the feasible region as to reach very robust and "high" quality results. However, for some problems in some runs, the approach was trapped in local optima solutions.

5 Conclusions and Future Work

We have presented a novel approach to solve engineering design problems using evolutionary algorithms. The approach does not use a penalty function to handle constraints. Instead, it has a mechanism to allow the closest solutions to the feasible region located in promising areas of the search space to remain in the population. The selection is

based on feasibility criteria and closeness to the feasible region. This mechanism does not add significant extra computational cost and it is very simple to implement. The approach was compared against penalty-function-based approaches and also against two state-of-the-art techniques providing a very competitive performance. Our future work consists on designing a mechanism to improve the local search capabilities of the approach in order to provide better results and also to apply it in the solution of dynamic/noisy optimization problems.

Acknowledgments. The first author acknowledges support from the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through a postdoctoral position at CINVESTAV-IPN's Electrical Engineering Department. The second author acknowledges support from (CONACyT) through project number 45683.

References

1. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer (2004)
2. Miettinen, K., Makela, M., Toivanen, J.: Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. Journal of Global Optimization **27** (2003) 427–446
3. Ray, T., Liew, K.: Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior. IEEE Transactions on Evolutionary Computation **7** (2003) 386–396
4. Hernández-Aguirre, A., Botello-Rionda, S., Coello Coello, C.A.: PASSSS: An Implementation of a Novel Diversity Strategy for Handling Constraints. In: Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004). Volume 1., Piscataway, New Jersey, Portland, Oregon, USA, IEEE Service Center (2004) 403–410
5. He, S., Prempain, E., Q.H.Wu: An Improved Particle Swarm Optimizer for Mechanical Design Optimization Problems. Engineering Optimization **36** (2004) 585–605
6. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
7. Schwefel, H.P.: Numerical Optimization of Computer Models. Wiley, England (1981)
8. Hoffmeister, F., Sprave, J.: Problem-independent handling of constraints by use of metric penalty functions. In Fogel, L.J., et al., eds.: Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96), San Diego, California, The MIT Press (1996) 289–294
9. Joines, J., Houck, C.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In Fogel, D., ed.: Proceedings of the first IEEE Conference on Evolutionary Computation, Orlando, Florida, IEEE Press (1994) 579–584
10. Hadj-Alouane, A.B., Bean, J.C.: A Genetic Algorithm for the Multiple-Choice Integer Program. Operations Research **45** (1997) 92–101
11. Mezura-Montes, E., Coello Coello, C.A.: Adding a Diversity Mechanism to a Simple Evolution Strategy to Solve Constrained Optimization Problems. In: Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003). Volume 1., Piscataway, New Jersey, Canberra, Australia, IEEE Service Center (2003) 6–13

Appendix

Full description of the four problems used in the experiments:

Problem 1: (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. There are four design variables as shown in Figure 3a: h (x_1), l (x_2), t (x_3) and b (x_4). The problem can be stated as follows:

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

Subject to:

$g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{max} \leq 0 \quad g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{max} \leq 0 \quad g_3(\boldsymbol{x}) = x_1 - x_4 \leq 0$
$g_4(\boldsymbol{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \quad g_5(\boldsymbol{x}) = 0.125 - x_1 \leq 0$
$g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{max} \leq 0 \quad g_7(\boldsymbol{x}) = P - P_c(\boldsymbol{x}) \leq 0$

where $\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$ $\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right)$
$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}$ $J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}$ $\sigma(\boldsymbol{x}) = \frac{6PL}{x_4x_3^2}, \delta(\mathbf{X}) = \frac{4PL^3}{Ex_3^3x_4}$

$P_c(\boldsymbol{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$ $P = 6000\ lb, \quad L = 14\ in, \quad E = 30 \times 10^6\ psi, \ G = 12{\times}10^6\ psi\ \tau_{max} = 13,600\,psi, \ \sigma_{max} = 30,000\,psi, \ \delta_{max} = 0.25\,in$
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 2: (Design of a Pressure Vessel) A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure 3b. 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. The problem can be stated as follows:

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

Subject to :

$g_1(\boldsymbol{x}) = -x_1 + 0.0193x_3 \leq 0 \quad g_2(\boldsymbol{x}) = -x_2 + 0.00954x_3 \leq 0$
$g_3(\boldsymbol{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0 \quad g_4(\boldsymbol{x}) = x_4 - 240 \leq 0$
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 3: (Minimization of the Weight of a Tension/Compression String) This problem consists of minimizing the weight of a tension/compression spring (see Figure 3c) 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:
$g_1(\boldsymbol{x}) = 1 - \frac{D^3 N}{71785d^4} \leq 0 \quad g_2(\boldsymbol{x}) = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \leq 0$
$g_3(\boldsymbol{x}) = 1 - \frac{140.45d}{D^2 N} \leq 0 \quad g_4(\boldsymbol{x}) = \frac{D+d}{1.5} - 1 \leq 0$
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 : $f(\boldsymbol{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$

Subject to :

$g_1(\boldsymbol{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0 \quad g_2(\boldsymbol{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0 \quad g_3(\boldsymbol{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$

$g_4(\boldsymbol{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0 \quad g_5(\boldsymbol{x}) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6\right)^{1/2}}{110.0x_6^3} - 1 \leq 0$

$g_6(\boldsymbol{x}) = \frac{\left(\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5\times10^6\right)^{1/2}}{85.0x_7^3} - 1 \leq 0 \quad g_7(\boldsymbol{x}) = \frac{x_2x_3}{40} - 1 \leq 0 \quad g_8(\boldsymbol{x}) = \frac{5x_2}{x_1} - 1 \leq 0$

$g_9(\boldsymbol{x}) = \frac{x_1}{12x_2} - 1 \leq 0 \quad g_{10}(\boldsymbol{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0 \quad g_{11}(\boldsymbol{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$

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 $5.0 \leq x_7 \leq 5.5$.