IS-PAES: A Constraint-Handling Technique Based on Multiobjective Optimization Concepts

Arturo Hernández Aguirre[1], Salvador Botello Rionda[1], Giovanni Lizárraga Lizárraga[1]
and Carlos A. Coello Coello[2]

[1] Center for Research in Mathematics (CIMAT)
Department of Computer Science
Guanajuato, Gto. 36240, MEXICO
artha,botello,giovanni@cimat.mx
[2] CINVESTAV-IPN
Evolutionary Computation Group
Depto. de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. 07300
ccoello@cs.cinvestav.mx

Abstract. This paper introduces a new constraint-handling method called Inverted-Shrinkable PAES (IS-PAES), which focuses the search effort of an evolutionary algorithm on specific areas of the feasible region by shrinking the constrained space of single-objective optimization problems. IS-PAES uses an adaptive grid as the original PAES (Pareto Archived Evolution Strategy). However, the adaptive grid of IS-PAES does not have the serious scalability problems of the original PAES. The proposed constraint-handling approach is validated with several examples taken from the standard literature on evolutionary optimization.

1 Introduction

Evolutionary Algorithms (EAs) in general (i.e., genetic algorithms, evolution strategies and evolutionary programming) lack a mechanism able to bias efficiently the search towards the feasible region in constrained search spaces. Such a mechanism is highly desirable since most real-world problems have constraints which could be of any type (equality, inequality, linear and nonlinear). The success of EAs in global optimization has triggered a considerable amount of research regarding the development of mechanisms able to incorporate information about the constraints of a problem into the fitness function of the EA used to optimize it [4, 17]. So far, the most common approach adopted in the evolutionary optimization literature 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 indicates the degree of penalization to be applied [4]. Recently, some researchers have suggested the use of multiobjective optimization concepts to handle constraints in EAs (see for example [4]). This paper introduces a new approach that is based on an evolution strategy that was originally proposed for multiobjective optimization: the Pareto Archived Evolution Strategy (PAES) [14]. Our approach can be used to handle constraints both of single- and multiobjective optimization problems and does not present the scalability problems of the original PAES. The remainder of this paper is organized as follows. Section 2 gives a formal description of the general problem that we want to solve. Section 3 describes the previous work related to our own. In Section 4, we describe the main algorithm of IS-PAES. Section 5 provides a comparison of results and Section 6 draws our conclusions and provides some paths of future research.

2 Problem Statement

We are interested in the general non-linear programming problem in which we want to:

$$\text{Find } \boldsymbol{x} \text{ which optimizes } f(\boldsymbol{x}) \tag{1}$$

subject to:

$$g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, n \tag{2}$$

$$h_j(\boldsymbol{x}) = 0, \quad j = 1, \ldots, p \tag{3}$$

where $\boldsymbol{x}$ is the vector of solutions $\boldsymbol{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or non-linear).

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 $\mathcal{F} \subseteq \mathcal{S}$.

For an inequality constaint that satisfies $g_i(\boldsymbol{x}) = 0$, then we will say that is active at $\boldsymbol{x}$. All equality constraints h_j (regardless of the value of $\boldsymbol{x}$ used) are considered active at all points of $\mathcal{F}$.

3 Related Work

Since our approach belongs to the group of techniques in which multiobjective optimization concepts are adopted to handle constraints, we will briefly discuss some of the most relevant work done in this area. The main idea of adopting multiobjective optimization concepts to handle constraints is to redefine the single-objective optimization of $f(\boldsymbol{x})$ as a multiobjective optimization problem in which we will have $m + 1$ objectives, where m is the total number of constraints. Then, we can apply any multiobjective optimization technique [9] to the new vector $\bar{v} = (f(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$, where $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are the original constraints of the problem. An ideal solution $\boldsymbol{x}$ would thus have $f_i(\boldsymbol{x})$=0 for $1 \leq i \leq m$ and $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ for all feasible $\boldsymbol{y}$ (assuming minimization).

Three are the mechanisms taken from evolutionary multiobjective optimization that are more frequently incorporated into constraint-handling techniques:

1. Use of Pareto dominance as a selection criterion.
2. Use of Pareto ranking [12] to assign fitness in such a way that nondominated individuals (i.e., feasible individuals in this case) are assigned a higher fitness value.
3. Split the population in subpopulations that are evaluated either with respect to the objective function or with respect to a single constraint of the problem. This is the selection mechanism adopted in the Vector Evaluated Genetic Algorithm (VEGA) [23].

We will now provide a brief discussion of the different approaches that have been proposed in the literature adopting the three main ideas previously indicated.

3.1 COMOGA

Surry & Radcliffe [24] used a combination of the Vector Evaluated Genetic Algorithm (VEGA) [23] and Pareto Ranking to handle constraints in an approach called COMOGA (Constrained Optimization by Multi-Objective Genetic Algorithms).

In this technique, individuals are ranked depending of their sum of constraint violation (number of individuals dominated by a solution). However, the selection process is based not only on ranks, but also on the fitness of each solution. COMOGA uses a non-generational GA and extra parameters defined by the user (e.g., a parameter called ϵ is used to define the change rate of P_{cost}). One of these parameters is P_{cost}, that sets the rate of selection based on fitness. The remaining $1 - P_{cost}$ individuals are selected based on ranking values. P_{cost} is defined by the user at the begining of the process and it is adapted during evolution using as a basis the percentage of feasible individuals that one wishes to have in the population.

COMOGA was applied on a gas network design problem and it was compared against a penalty function approach. Although COMOGA showed a slight improvement in the results with respect to a penalty function, its main advantage is that it does not requiere a fine tuning of penalty factors or any other additional parameter. The main drawback of COMOGA is that it requires several extra parameters, although its authors argue that the technique is not particularly sensitive to their values [24].

3.2 VEGA

Parmee & Purchase [18] proposed to use VEGA [23] to guide the search of an evolutionary algorithm to the feasible region of an optimal gas turbine design problem with a heavily constrained search space. After having a feasible point, they generated an optimal hypercube around it in order to avoid leaving the feasible region after applying the genetic operators. Note that this approach does not really use Pareto dominance or any other multiobjective optimization concepts to exploit the search space. Instead, it uses VEGA just to reach the feasible region. The use of special operators that preserve feasibility makes this approach highly specific to one application domain rather than providing a general methodology to handle constraints.

Coello [7] used a population-based approach similar to VEGA [23] to handle constraints in single-objective optimization problems. At each generation, the population was split into $m+1$ subpopulations of equal fixed size, where m is the number of constraints of the problem. The additional subpopulation handles the objective function of the problem and the individuals contained within it are selected based on the unconstrained objective function value. The m remaining subpopulations take one constraint of the problem each as their fitness function. The aim is that each of the subpopulations tries to reach the feasible region corresponding to one individual constraint. By combining these different subpopulations, the approach will reach the feasible region of the problem considering all of its constraints simultaneously.

This approach was tested with some engineering problems [7] in which it produced competitive results. It has also been successfully used to solve combinational circuit design problems [8]. The main drawback of this approach is that the number of subpopulations required increases linearly with the number of constraints of the problem. This has some obvious scalability problems. Furthermore, it is not clear how to determine appropriate sizes for each of the subpopulations used.

3.3 MOGA

Coello [6] proposed the use of Pareto dominance selection to handle constraints in EAs. This is an application of Fonseca and Fleming's Pareto ranking process [11] (called Multi-Objective Genetic Algorithm, or MOGA) to constraint-handling. In this approach, feasible individuals are always ranked higher than infeasible ones. Based on this rank, a fitness value is assigned to each individual. This technique also includes a self-adaptation mechanism that avoids the usual empirical fine-tuning of the main genetic operators. Coello's approach uses a real-coded GA with universal stochastic sampling selection (to reduce the selection pressure caused by the Pareto ranking process).

This approach has been used to solve some engineering design problems [6] in which it produced very good results. Furthermore, the approach showed great robustness and required a relatively low number of fitness function evaluations with respect to traditional penalty functions. Additionally, it does not require any extra parameters. Its main drawback is the computational cost ($O(M^2)$, where M is the population size) derived from the Pareto ranking process.

3.4 NPGA

Coello and Mezura [5] 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 NPGA, Coello and Mezura's approach does not require niches (or fitness sharing [10]) to maintain diversity in the population. The NPGA is a more efficient technique than traditional multiobjective optimization algorithms, since it does not compare every individual in the population with respect to each other (as in traditional Pareto ranking), but uses only a sample of the population to estimate Pareto dominance. This is the main advantage of this approach with respect to Coello's proposal [6]. Note however that Coello and Mezura's approach requires an additional parameter called S_r that controls the diversity of the population. S_r indicates the proportion of parents selected by four comparison criteria described below. The remaining $1 - S_r$ parents will be selected by a pure probabilistic approach. Thus, this mechanism is responsible for keeping infeasible individuals in the population (i.e., the source of diversity that keeps the algorithm from converging to a local optimum too early in the evolutionary process).

This approach has been tested with several benchmark problems and was compared against several types of penalty functions. Results indicated that the approach was robust, efficient and effective. However, it was also found that the approach had scalability problems (its performance degrades as the number of decision variables increases).

3.5 Pareto Set and Line Search

Camponogara & Talukdar [3] proposed an approach in which a global optimization problem was transformed into a bi-objective problem where the first objective is to optimize the original objective function and the second is to minimize:

$$\Phi(\mathbf{x}) = \sum_{i=1}^{n} \max(0, g_i(\mathbf{x})) \tag{4}$$

Equation (4) tries to minimize the total amount of constraint violation of a solution (i.e., it tries to make it feasible). At each generation of the process, several Pareto sets are generated. An operator that substitutes crossover takes two Pareto sets S_i and S_j, where $i < j$, and two solutions $x_i \in S_i$ and $x_j \in S_j$, where x_i dominates x_j. With these two points a search direction is defined using:

$$d = \frac{(x_i - x_j)}{|x_i - x_j|} \tag{5}$$

Line search begins by projecting d over one variable axis on decision variable space in order to find a new solution x which dominates both x_i and x_j. At pre-defined intervals, the worst half of the population is replaced with new random solutions to avoid premature convergence. This indicates some of the problems of the approach to maintain diversity. Additionally, the use of line search within a GA adds some extra computational cost.

The authors of this approach validated it using a benchmark consisting of five test functions. The results obtained were either optimal or very close to it. The main drawback of this approach is its additional computational cost. Also, it is not clear what is the impact of the segment chosen to search on the overall performance of the algorithm.

3.6 Pareto Ranking and Domain Knowledge

Ray et al. [19] proposed the use of a Pareto ranking approach that operates on three spaces: objective space, constraint space and the combination of the two previous spaces. This approach also uses mating restrictions to ensure better constraint satisfaction in the offspring generated and a selection process that eliminates weaknesses in any of these spaces. To maintain diversity, a niche mechanism based on Euclidean distances is used.

This approach can solve both constrained or unconstrained optimization problems with one or several objective functions.

The main advantage of this approach is that it requires a very low number of fitness function evaluations (between 2% and 10% of the number of evaluations required by the homomorphous maps of Koziel and Michalewicz [15], which is one of the best constraint-handling techniques known to date). The technique has some problems to reach the global optima, but it produces very good approximations considering its low computation cost. The main drawback of the approach is that its implementation is considerably more complex than any of the other techniques previously discussed.

3.7 Pareto Ranking and Robust Optimization

Ray [20] explored an extension of his previous work on constraint-handling [19] in which the emphasis was robustness. A robust optimized solution is not sensitive to parametric variations due to incomplete information of the problem or to changes on it. This approach is capable of handling constraints and finds feasible solutions that are robust to parametric variations produced over time. This is achieved using the individual's self-feasibility and its neighborhood feasibility. The results reported in two well-known design problems [20] showed that the proposed approach did not reach solutions as good as the other techniques with which it was compared, but it turned out to be less sensitive to parametric variations, which was the main goal of the approach. In contrast, the other techniques analyzed showed significant changes when the parameters were perturbed. The main drawback of this approach is, again, its relative complexity (i.e., its difficulty to implement it), and it would also be desirable that the approach is further refined so that it can get closer to the global optimum than the current available version.

4 IS-PAES Algorithm

All of the approaches discussed in the previous section have drawbacks that keep them from producing competitive results with respect to the constraint-handling techniques that represent the state-of-the-art in evolutionary optimization. In a recent technical report [16], four of the previous techniques (i.e., COMOGA [24], VEGA [7], MOGA [6] and NPGA [5]) have been compared using Michalewicz's benchmark [17] and some additional engineering optimization problems. Although inconclusive, the results indicate that the use of Pareto dominance as a selection criterion gives better results than Pareto ranking or the use of a population-based approach. However, in all cases, the approaches analyzed are unable to reach the global optimum of problems with either high dimensionality, large feasible regions or many nonlinear equality constraints [16].

In contrast, the approach proposed in this paper uses Pareto dominance as the criterion selection, but unlike the previous work in the area, a secondary population is used in this case. The approach, which is a relatively simple extension of PAES [14] provides, however, very good results, which are highly competitive with those generated with an approach that represents the state-of-the-art in constrained evolutionary optimization.

IS-PAES has been implemented as an extension of the Pareto Archived Evolution Strategy (PAES) proposed by Knowles and Corne [14] for multiobjective optimization. PAES's main feature is the use of an adaptive grid on which objective function space is located using a coordinate system. Such a grid is the diversity maintenance mechanism of PAES and it's the main feature of this algorithm. The grid is created by bisecting k times the function space of dimension $d = g+1$. The control of 2^{kd} grid cells means the allocation of a large amount of physical memory for even small problems. For instance, 10 functions and 5 bisections of the space produce 2^{50} cells. Thus, the first feature introduced in IS-PAES is the "inverted" part of the algorithm that deals with this space usage problem. IS-PAES's fitness function is mainly driven by a feasibility criterion. Global information carried by the individuals surrounding the feasible region is used to concentrate the search effort on smaller areas as the evolutionary process takes place. In consequence, the search space being explored is "shrunk" over time. Eventually, upon termination, the size of the search space being inspected will be very small and will contain the solution desired (in the case of single-objective problems. For multi-objective problems, it will contain the feasible region). The main algorithm of IS-PAES is shown in Figure 1.

```
maxsize: max size of file
$c$: current parent $\in X$ (decision variable space)
$h$:child of c $\in X$, $a_h$: individual in file that dominates h
$a_d$: individual in file dominated by h
$current$: current number of individuals in file
cnew: number of individuals generated thus far
$current$ = 1; cnew=0; c = newindividual() ; add(c)
While cnew$\leq$MaxNew do
    h = mutate(c); cnew$+=$1;
    if (c$\preceq$h) then exit loop
    else if (h$\preceq$c) then { remove(c); add(g); c=h; }
    else if ($\exists$ a$_h$ $\in$ file | a$_h$ $\preceq$ h) then exit loop
    else if ($\exists$ a$_d$ $\in$ file | h $\preceq$ a$_d$) then
        add( h ); $\forall$ a$_d$ { remove(a$_d$); current$-=$1 }
    else test(h,c,file)
    if (cnew % g==0) then c = individual in less densely populated region
    if (cnew % r==0) then shrinkspace(file)
End While
```

Fig. 1. Main algorithm of IS-PAES

The function **test(h,c,file)** determines if an individual can be added to the external memory or not. Here we introduce the following notation: $x_1 \Box x_2$ means x_1 is located in a less populated region of the grid than x_2. The pseudo-code of this function is depicted in Figure 2.

```
if (current $<$ maxsize) then add(h)
    if (h □ c) then c = h
else if ($\exists a_p \in$file | h □ $a_p$) then { remove($a_p$); add(h) }
if (h □ c) then c = h;
```

Fig. 2. Pseudo-code of **test(h,c,file)**

4.1 Inverted “ownership”

IS-PAES handles the population *as part of* a grid location relationship, whereas PAES handles a grid location *contains* population relationship. In other words, PAES keeps a list of individuals on either grid location, but in IS-PAES either individual knows its position on the grid. Therefore, building a sorted list of the most dense populated areas of the grid only requires to sort the k elements of the external memory. In PAES, this procedure needs to inspect every location of the grid in order to produce an unsorted list, there after the list is sorted. The advantage of the inverted relationship is clear when the optimization problem has many functions (more than 10), and/or the granularity of the grid is fine, for in this case only IS-PAES is able to deal with any number of functions and granularity level.

4.2 Shrinking the objective space

```
$\underline{\boldsymbol{x}}_{pob}$: vector containing the smallest value of either $x_i \in X$
$\overline{\boldsymbol{x}}_{pob}$: vector containing the largest value of either $x_i \in X$
select(file); getMinMax( file, $\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$)
trim($\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$ )
adjustparameters(file);
```

Fig. 3. Pseudo-code of **Shrinkspace(file)**

Shrinkspace(file) is the most important function of IS-PAES since its task is the reduction of the search space. The pseudo-code of **Shrinkspace(file)** is shown in Figure 3.

The function **select(file)** returns a list whose elements are the best individuals found in *file*. The size of the list is 15% of *maxsize*. Since individuals could be feasible, infeasible or only partially feasible, the list is generated by discarding from the file the worst elements based on constraint violation. Notice that **select(file)** does not use a greedy approach (e.g., searching for the best feasible individuals at once). Instead, individuals with the highest amount of constraint violation are removed from the file. Thus, the resulting list contains: 1) only the best feasible individuals, 2) a combination of feasible and partially feasible individuals, or 3) the "best" infeasible individuals. Function **trim(**$\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$**)** shrinks the feasible space around the potential solutions enclosed in the hypervolume defined by the vectors $\underline{\boldsymbol{x}}_{pob}$ and $\overline{\boldsymbol{x}}_{pob}$. Thus, the function **trim()** (see Figure 4) determines the new boundaries for the decision variables.

```
n: size of decision vector;
$\overline{x}_i$: actual upper bound of the $i_{th}$ decision variable
$\underline{x}_i$: actual lower bound of the $i_{th}$ decision variable
$\overline{x}_{pob,i}$: upper bound of $i_{th}$ decision variable in population
$\underline{x}_{pob,i}$: lower bound of $i_{th}$ decision variable in population
$\forall i$ : i ∈ { 1, ..., n }
    $slack_i = 0.05 \times (\overline{x}_{pob,i} - \underline{x}_{pob,i})$
    $width\_pob_i = \overline{x}_{pob,i} - \underline{x}_{pob,i}$; $width^t_i = \overline{x}^t_i - \underline{x}^t_i$
    $deltaMin_i = \frac{\beta * width^t_i - width\_pob_i}{2}$
    delta$_i$ = max(slack$_i$, deltaMin$_i$);
    $\overline{x}^{t+1}_i = \overline{x}_{pob,i} + delta_i$; $\underline{x}^{t+1}_i = \underline{x}_{pob,i} - delta_i$;
    if ($\overline{x}^{t+1}_i > \overline{x}_{original,i}$) then
        $\underline{x}^{t+1}_i - = \overline{x}^{t+1}_i - \overline{x}_{original,i}$; $\overline{x}^{t+1}_i = \overline{x}_{original,i}$;
    if ($\underline{x}^{t+1}_i < x_{original,i}$) then $\overline{x}^{t+1}_i + = \underline{x}_{original,i} - \underline{x}^{t+1}_i$;
        $\underline{x}^{t+1}_i = \underline{x}_{original,i}$;
    if ($\overline{x}^{t+1} > \overline{x}_{original,i}$) then $\overline{x}^{t+1}_i = \overline{x}_{original,i}$;
```

Fig. 4. Pseudo-code of **trim**

The value of β is the percentage by which the boundary values of either $x_i \in X$ must be reduced such that the resulting hypervolume is a fraction α of its initial value. In our experiments, $\alpha = 0.90$ worked well in all cases. Clearly, α controls the shrinking speed, hence the algorithm is sensitive to this parameter and it can prevent it from finding the optimum solution if small values are chosen. In our experiments, values in the range $[85\%, 95\%]$ were tested with no visible effect in the performance. Of course, α values near to 100% slow down the convergence speed. The last step of **shrinkspace()** is a call to **adjustparameters(file)**. The goal is to re-start the control variable σ using: $\sigma_i = (\overline{x}_i - \underline{x}_i)/\sqrt{n} \quad i \in (1, \ldots, n)$ This expression is also used during the generation of the initial population. In that case, the upper and lower bounds take the initial values of the search space indicated by the problem. The variation of the mutation probability follows the exponential behavior suggested by Bäck [1].

5 Comparison of Results

We have validated our approach with several problems used as a benchmark for evolutionary algorithms (see [17]) and with several engineering optimization problems taken from the standard literature. In the first case, our results are compared against a technique called "stochastic ranking" [22], which is representative of the state-of-the-art in constrained evolutionary optimization. This approach has been found to be equally good or even better in some cases than the homomorphous maps of Koziel and Michalewicz [15].

5.1 Examples

The following parameters were adopted for IS-PAES in all the experiments reported next: $maxsize = 200$, $bestindividuals = 15\%$, $slack = 0.05$, $r = 400$. The maximum number of fitness function evaluations was set to 350,000, which is the number of evaluations used in [22]. We used ten (out of 13) of the test functions described in [22], due to time limitations to perform the experiments. The test functions chosen, however, contain characteristics that are representative of what can be considered "difficult" global optimization problems for an evolutionary algorithm.

TF	**n**	**Type of function**	ρ	**LI**	**NI**	**NE**
g01	13	quadratic	0.0003%	9	0	0
g02	20	non linear	99.9973%	2	0	0
g03	10	non linear	0.0026%	0	0	1
g04	5	quadratic	27.0079%	4	2	0
g06	2	non linear	0.0057%	0	2	0
g07	10	quadratic	0.0000%	3	5	0
g08	2	non linear	0.8581%	0	2	0
g09	7	non linear	0.5199%	0	4	0
g10	8	linear	0.0020%	6	0	0
g11	2	quadratic	0.0973%	0	0	1

Table 1. Values of ρ for the ten test problems chosen.

To get a better idea of the difficulty of solving each of these problems, a ρ metric (as suggested by Koziel and Michalewicz [15]) was computed using the following expression:

$$\rho = |F|/|S| \tag{6}$$

where $|F|$ is the number of feasible solutions and $|S|$ is the total number of solutions randomly generated. In this work, we generated $S = 1,000,000$ random solutions. The different values of ρ for each of the test functions chosen are shown in Table 1, where n is the number of decision variables, LI is the number of linear inequalities, NI the number of nonlinear inequalities and NE is the number of nonlinear equalities.

From Tables 2 and 3 we can see that the proposed approach is highly competitive. The discussion of results for each test function is provided next:

For **g01** the best solution found by IS-PAES was: $\mathbf{x} = \{1$, 0.999999939809, 0.999997901977, 1, 0.999981406123, 1, 1, 0.999999242667, 0.999981194574,

TF	optimal	Best	Mean	Median	Worst	Std Dev
g01	-15.0	-14.9997	-14.494	-14.997	-12.446	9.3×10^{-1}
g02	-0.803619	-0.803376	-0.793281	-0.793342	-0.768291	9.0×10^{-3}
g03	-1.0	-1.000	-1.000	-1.000	-1.000	9.7×10^{-5}
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	0.0
g06	-6961.814	-6961.814	-6961.813	-6961.814	-6961.810	8.5×10^{-5}
g07	24.306	24.338	24.527	24.467	24.995	1.7×10^{-1}
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	0.0
g09	680.630	680.630	680.631	680.631	680.634	8.1×10^{-4}
g10	7049.331	7062.019	7342.944	7448.014	7588.054	1.4×10^{2}
g11	0.750	0.750	0.750	0.750	0.751	2.6×10^{-4}

Table 2. Results produced by our IS-PAES algorithm.

2.99987534752, 2.99995011286, 2.99993014684, 0.999982112914$\}$ with $F(\mathbf{x}) = -14.99968877$. In this case, IS-PAES was less consistent than stochastic ranking in finding the global optimum, mainly because the approach was trapped in a local optimum in which $F(\mathbf{x}) = -13$ during 20% of the runs.

TF	optimal	Best	Mean	Median	Worst	Std Dev
g01	-15.0	-15.0	-15.0	-15.0	-15.0	0.0
g02	-0.803619	-0.803515	-0.781975	-0.785800	-0.726288	2×10^{-2}
g03	-1.0	-1.000	-1.000	-1.000	-1.000	1.9×10^{-4}
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	2.0×10^{-5}
g06	-6961.814	-6961.814	-6875.940	-6961.814	-6350.262	1.6×10^{2}
g07	24.306	24.307	24.374	24.357	24.642	6.6×10^{-2}
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	2.6×10^{-17}
g09	680.630	680.630	680.656	680.641	680.763	3.4×10^{-2}
g10	7049.331	7054.316	7559.192	7372.613	8835.655	5.3×10^{2}
g11	0.750	0.750	0.750	0.750	0.750	8.0×10^{-5}

Table 3. Results produced by the stochastic ranking algorithm [22].

For **g02** the best solution found by IS-PAES was: $\mathbf{x} = \{$3.14860401788, 3.10915903011, 3.08909341555, 3.05835689132, 3.04000196011, 3.00100530894, 2.94955289769, 2.94207158769, 0.49907406319, 0.486231653274, 0.49055938302, 0.492879188045, 0.481722447567, 0.471623533316, 0.452037376504, 0.442565813637, 0.451211591495, 0.437863945589, 0.444359423833, 0.437834075871$\}$ with $F(\mathbf{x}) = -0.803375563$. As we can see, the best result found by stochastic ranking was better than the best result found by IS-PAES. However, the statistical performance measures of IS-PAES were better (particularly the standard deviation which is significantly lower), which seems to indicate that our approach had more robustness in this problem.

The best solution found by IS-PAES for **g03** was: $\mathbf{x} = \{$0.316965968, 0.315664596, 0.314608242, 0.315958975, 0.315915392, 0.317873891, 0.316867036, 0.314518512, 0.314381436, 0.319636209$\}$ with $F(\mathbf{x}) = -1.000421429$. In can be clearly seen in this case that both IS-PAES and stochastic ranking had an excellent performance.

The best solution found by IS-PAES for **g04** was: $\mathbf{x}$ = $\{78, 33.00000002, 29.99525605,$ 45, 36.77581285 $\}$ with $F(\mathbf{x}) = -30665.53867$. The behavior of IS-PAES in this test function was practically the same as stochastic ranking.

For **g06**, the best solution found by IS-PAES was: $\mathbf{x}$ = $\{$14.0950000092, 0.842960808844$\}$ with $F(\mathbf{x})$ = -6961.813854. Note that both approaches reach the global optimum in this case, but IS-PAES is more consistent, with very small variations in the results and a much lower standard deviation than stochastic ranking.

Stochastic ranking was clearly better in all aspects than IS-PAES for **g07**. The best solution found by IS-PAES was: $\mathbf{x}$ = $\{$2.16996489702, 2.36701436984, 8.76882720318, 5.07418756668, 0.943992761955, 1.32027308617, 1.31870032997, 9.82673763033, 8.26988778617, 8.36187863755$\}$ with $F(\mathbf{x}) = 24.33817628$.

For **g08**, the best solution found by IS-PAES was: $\mathbf{x}$ = $\{$1.227971353, 4.245373368$\}$ with $F(\mathbf{x}) = -0.095825041$. Both algorithms had the same performance in this test function.

Both algorithms reached the global optimum for **g09**, but IS-PAES had better statistical measures. The best solution found by IS-PAEs was: $\mathbf{x}$ = $\{$2.326603718, 1.957743917, -0.468352679, 4.349668424, -0.621354832, 1.047882344, 1.590801921$\}$ with $F(\mathbf{x}) = 680.6304707$.

Except for the best solution found (which is better for stochastic ranking), the statistical measures of IS-PAES are better than those of stochastic ranking for **g10**. The best solution found by IS-PAEs was: $\mathbf{x}$ = $\{$105.6345328, 1179.227593, 6070.09281, 122.497943, 257.1979828, 277.4889774, 265.2967614, 357.197398$\}$ with $F(\mathbf{x}) = 7062.019117$.

Finally, for **g11** both algorithms had a very good performance. The best solution found by IS-PAES was: $\mathbf{x}$ = $\{$2.326603718, 1.957743917, -0.468352679, 4.349668424, -0.621354832, 1.047882344, 1.590801921$\}$ with $F(\mathbf{x}) = 0.749913273$.

5.2 Optimization of a 49-bar plane truss

The engineering optimization problem chosen is the optimization of the 49-bar plane truss shown in Figure 5. The goal is to find the cross-sectional area of each member of the truss, such that the overall weight is minimized, subject to stress and displacement constraints. The weight of the truss is given by $F(\mathbf{x}) = \sum_{j=1}^{49} \gamma A_j L_j$, where A_j is the cross-sectional area of the j_{th} member, L_j is the corresponding length of the bar, and γ is the volumetric density of the material. We used a catalog of *Altos Hornos de México, S.A.*, with 65 entries for the cross-sectional areas available for the design. Other relevant data are the following: Young modulus $= 2.1 \cdot 10^6$ kg/cm^3, maximum allowable stress = 3500.00 kg/cm^2, $\gamma = 7.4250 \cdot 10^{-3}$, and a horizontal load of 4994.00 kg applied to the nodes: 3, 5, 7, 9, 12, 14, 16, 19, 21, 23, 25 y 27. We solved this problem for two cases:

1. **Case 1. Stress and displacement constraints:** Maximum allowable stress = 3500.00 kg/cm^2, maximum displacement per node $= 10cm$ A total of 72 constraints, thus 73 objective functions.

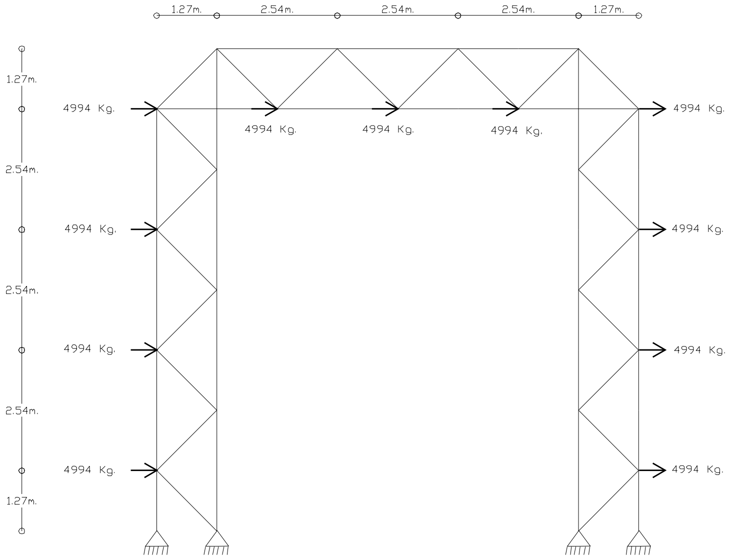

Fig. 5. 49-bar plane truss used as an engineering optimization example.

2. **Case 2. Real-world problem:** The design problem considers traction and compression stress on the bars, as well as their proper weight. Maximum allowable stress = 3500.00 kg/cm^2, maximum displacement per node =10 cm. A total of 72 constraints, thus 73 objective functions.

The average result of 30 runs for each case are shown in Table 4. We compare IS-PAES with previous results reported by Botello [2] using other heuristics with a penalty function [21] (SA: Simulated Annealing, GA50: Genetic Algorithm with a population of 50, and GSSA: General Stochastic Search Algorithm with populations of 50 and 5).

We can see in this case that IS-PAES produced the lowest average weight for CASE 1, and the second best for CASE 2.

Algorithm	CASE 1: Avg. Weight (Kg)	CASE 2: Avg. Weight (Kg)
IS-PAES	725	2603
SA	737	2724
GA50	817	2784
GSSA50	748	2570
GSSA5	769	2716

Table 4. Comparison of different algorithms on the 49-bar struss, cases 1 and 2.

6 Conclusions and Future Work

We have introduced a constraint-handling approach that combines multiobjective optimization concepts with an efficient reduction mechanism of the search space and a secondary population. We have shown how our approach overcomes the scalability problem of the original PAES from which it was derived, and we also showed that the approach is highly competitive with respect to the state-of-the-art technique in the area.

As part of our future work, we want to refine the mechanism adopted for reducing the search space being explored, since in our current version of the algorithm, convergence to local optima may occur in some cases due to the high selection pressure introduced by such mechanism.

The elimination of the parameters required by our approach is another goal of our current research. Finally, we also intend to couple the mechanisms proposed in this paper to other evolutionary multiobjective optimization approaches.

Acknowledgments

The first author acknowledges support from CONACyT project I-39324-A. The second author acknowledges support from CONACyT project No. 34575-A. The last author acknowledges support from NSF-CONACyT project No. 32999-A.

References

1. Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.
2. Salvador Botello, José Luis Marroquín, Eugenio Oñate, and Johan Van Horebeek. Solving Structural Optimization problems with Genetic Algorithms and Simulated Annealing. *International Journal for Numerical Methods in Engineering*, 45(8):1069–1084, July 1999.
3. Eduardo Camponogara and Sarosh N. Talukdar. A Genetic Algorithm for Constrained and Multiobjective Optimization. In Jarmo T. Alander, editor, *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, pages 49–62, Vaasa, Finland, August 1997. University of Vaasa.
4. Carlos 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.
5. Carlos A. Coello Coello and Efrén Mezura-Montes. Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments. In I.C. 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.
6. Carlos A. Coello Coello. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, 17:319–346, 2000.
7. Carlos A. Coello Coello. Treating Constraints as Objectives for Single-Objective Evolutionary Optimization. *Engineering Optimization*, 32(3):275–308, 2000.
8. Carlos A. Coello Coello and Arturo Hernández Aguirre. Design of Combinational Logic Circuits through an Evolutionary Multiobjective Optimization Approach. *Artificial Intelligence for Engineering, Design, Analysis and Manufacture*, 16(1):39–53, 2002.

9. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.
10. Kalyanmoy Deb and David E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In J. David 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.
11. Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.
12. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
13. Jeffrey Horn, Nicholas Nafpliotis, and David 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. Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
15. Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
16. Efrén Mezura-Montes and Carlos A. Coello Coello. A Numerical Comparison of some Multiobjective-based Techniques to Handle Constraints in Genetic Algorithms. Technical Report EVOCINV-03-2002, Evolutionary Computation Group at CINVESTAV-IPN, México, D.F. 07300, September 2002. available at: `http://www.cs.cinvestav.mx/˜EVOCINV/`.
17. Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.
18. I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pages 97–102, Plymouth, UK, 1994. University of Plymouth, University of Plymouth.
19. Tapabrata Ray, Tai Kang, and Seow Kian Chye. An Evolutionary Algorithm for Constrained Optimization. In Darrell Whitley et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, 2000. Morgan Kaufmann.
20. Tapabrata Ray and K.M. Liew. A Swarm Metaphor for Multiobjective Design Optimization. *Engineering Optimization*, 34(2):141–153, March 2002.
21. 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 (ICGA-89)*, pages 191–197, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.
22. T.P. Runarsson and X. Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.
23. J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.
24. Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3):391–412, 1997.