PASSSS: An Implementation of a Novel Diversity Strategy for Handling Constraints

Arturo Hernández Aguirre, Salvador Botello Rionda
Center for Research in Mathematics (CIMAT)
Department of Computer Science
A.P. 402, Guanajuato, Gto. C.P. 36000 MEXICO
Email: artha,botello@cimat.mx

Carlos A. Coello Coello
CINVESTAV-IPN
Computer Science Section, Electrical Eng. Dept.
México, D.F., C.P. 07300 MEXICO
Email: ccoello@cs.cinvestav.mx

***Abstract*— In this paper, we introduce PASSSS (PAS^4), the Pareto Archived and dominance Selection with Shrinkable Search Space evolutionary computation algorithm. The main contribution of this paper is a diversity control mechanism embedded into the selection operator of an evolutionary algorithm that can be used (with little or no modification) to solve both single-objective and multi-objective optimization problems. We present a detailed description of the PAS^4 algorithm, and illustrate its capabilities by solving several engineering design problems and some test functions from a well-known benchmark in evolutionary optimization. Additionally, PAS^4 is also used to solve continuous and discrete multiobjective engineering optimization problems.**

I. Introduction

The design of constraint-handling mechanisms for evolutionary algorithms has been the subject of considerable research in the last few years [12], [3]. Perhaps the main lesson learned from this research is that a key aspect when designing a constraint-handling mechanism is to be able to maintain diversity in the population of the evolutionary algorithm adopted for optimization [15], [7]. In this paper, we introduce a new selection operator that focuses on diversity. The PAS^4 algorithm uses a selection mechanism based on Pareto dominance [4]. However, unlike other constraint-handling approaches that separate constraints and objectives (see for example [5]), PAS^4 does not perform tournaments based on feasibility. PAS^4 inherits its main structure from the ISPAES algorithm introduced in [8], but it uses a $(\mu+\lambda)$-ES, enhanced with: an external memory, an adaptive grid that operates on objective function space (which helps to control diversity, like in PAES [9]), a mechanism to trim the search space and to focus the search effort on promising regions, and the selection mechanism introduced in this paper, and which we call "most promising selection" (called **test** in Figures 1 and 3).

The organization of the paper is the following: Section II introduces the formalities of the problem of interest to us. Section III describes the criteria used to establish Pareto dominance. Section IV explains how multiobjective concepts have been used to handle constraints in the specialized literature. Section V gives an introduction to the desirable properties of the selection procedure attached to a constraint-handling technique. Section VI presents a detailed description of the PAS^4 algorithm (for continuous search spaces), and the (simple) changes needed for solving discrete problems with the same approach. Section VII describes some engineering optimization problems taken from the standard literature. Finally, Section VIII draws our conclusions and provides some paths of future research.

II. Problem Statement

We are interested in the general nonlinear programming problem in which we want to:

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

subject to:

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

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

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 non-linear).

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

III. Basic Concepts

A multiobjective optimization problem (MOP) has the following form:

$$\text{minimize} \quad [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{4}$$

subject to the n inequality constraints:

$$g_i(\vec{x}) \leq 0 \qquad i = 1, 2, \ldots, n \tag{5}$$

and the p equality constraints:

$$h_j(\vec{x}) = 0 \qquad j = 1, 2, \ldots, p \tag{6}$$

where k is the number of objective functions $f_i : R^n \rightarrow R$. We call $\vec{x} = [x_1, x_2, \ldots, x_r]^T$ to the vector of decision variables. We wish to determine from among the set $\mathcal{F}$ of all vectors which satisfy (5) and (6) the particular set of

values $x_1^*, x_2^*, \ldots, x_n^*$ which yield the optimum values of all the objective functions.

A. Pareto Optimality

For a given multiobjective optimization problem, $\vec{f}(\vec{x})$, the *Pareto optimal set* ($\mathcal{P}^*$) is defined as:

$$\mathcal{P}^* := \{\vec{x} \in \mathcal{F} \mid \neg\exists\ \vec{x}' \in \mathcal{F}\ \ \vec{f}(\vec{x}') \preceq \vec{f}(\vec{x})\}. \qquad (7)$$

In words, this definition says that $\vec{x}^*$ is Pareto optimal if there exists no feasible vector of decision variables $\vec{x} \in \mathcal{F}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. Unfortunately, this concept almost always gives not a single solution, but rather a set of solutions called the *Pareto optimal set*. The vectors $\vec{x}^*$ corresponding to the solutions included in the Pareto optimal set are called *nondominated*. The image of the Pareto optimal set under the objective functions is called *Pareto front*.

IV. Related Work

The PAS^4 algorithm belongs to the group of techniques in which multiobjective optimization concepts are adopted to handle constraints. Such approaches normally adopt a redefinition of a single-objective optimization problem $f(\vec{x})$ as a multiobjective optimization one, in which we will have $m+1$ objectives, where m is the total number of constraints. Then, we can apply any multiobjective optimization technique [4] to the new vector $\bar{v} = (f(\vec{x}), f_1(\vec{x}), \ldots, f_m(\vec{x}))$, where $f_1(\vec{x}), \ldots, f_m(\vec{x})$ are the original constraints of the problem. An ideal solution $\vec{x}$ would thus have $f_i(\vec{x}) \geq 0$ for $1 \leq i \leq m$ and $f(\vec{x}) \leq f(\vec{y})$ for all feasible $\vec{y}$ (assuming minimization). Additionally to this redefinition of a single-objective optimization problem into a multiobjective optimization problem, PAS^4 uses a selection mechanism based on Pareto dominance and retains the nondominated solutions found along the evolutionary process in an external archive.

Other authors have adopted selection schemes based on Pareto dominance (or Pareto ranking) (see for example [17]). However, none of these authors adopted an external archive to retain nondominated solutions, and all of these previous approaches presented problems related to loss of diversity [3]. Additionally, no effort was made on any of these previous approaches for focusing the search effort towards the most promising regions of the space explored so far. These are the main issues that distinguishes our proposal from the previous work reported in the literature.

V. Selection operator

Several other authors have identified in the past that, in order to sample the feasible region of the search space widely enough as to reach the global optimum, it is necessary to maintain a balance between feasible and infeasible solutions [7], [11]. In this paper, the desired population composition produced by an "ideal" selection operator is called *selection blend*. Also, it is desirable that the selection operator is able to maintain *diversity* in the population, which in our case implies keeping both feasible and infeasible solutions in the population. These goals are difficult to achieve when the selection operator is driven by "greedy rules" which tend to be incompatible with our goal of maintaining infeasible solutions in the population. For example, it could be the case that the selection operator tends to increase the number of feasible solutions over time, always disregarding infeasible individuals. Such an operator opposes our goal of maintaining diversity in the population. The proposal in PAS^4 is to adopt a selection operator based on Pareto dominance (see Figure 1), which performs the following tasks: first, it finds *the* individual in the external population (i.e., our approximation of the Pareto optimal set found so far) with the maximum amount of violation with respect to the first constraint, and it deletes it. Then, we consider only the second constraint, and we apply the same procedure. This process is repeated until we have traversed all the constraints of the problem. Note that since we do not delete individuals based on their total amount of constraint violation (or closeness to the feasible region), the operator is able to preserve both feasible and infeasible solutions in the population.

PAS^4 ALGORITHM

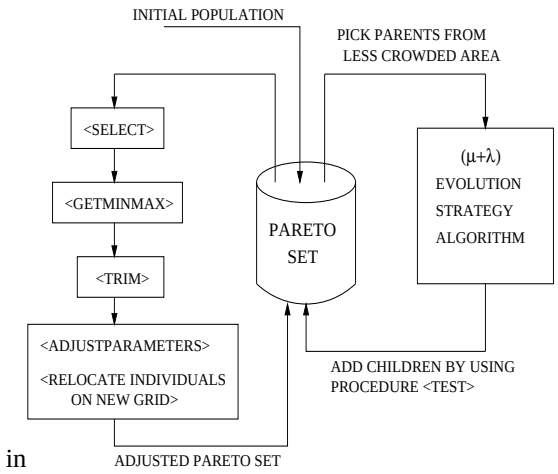

Fig. 1. The logical structure of the PAS^4 algorithm.

VI. The PAS^4 Algorithm

The approach proposed in this paper adopts an external population that keeps the nondominated solutions found along the evolutionary process. An *adaptive grid* applied over objective functions space is used to keep diversity as in the Pareto Archived Evolution Strategy (PAES) [9]. However, in PAS^4 the grid reduces its size over time (i.e., generations). Thus, its spread decreases over time, focusing the search effort on the most promising regions of the search space. The logical structure of PAS^4 is shown in Figure 1. Note the two loops operating over the external population (or memory). The right loop aims to explore the search space, while the left loop aims to keep population diversity and to perform an exploitation of solutions.

```
maxsize: max size of Pareto store
maxffeval: fitness function evaluations
Initialize Pareto store with maxsize individuals
While gen≤MaxGen do
    Pick μ parents from less crowded area
    Run (μ + λ)-ES until maxffeval is met
    test(Pareto store, λ children)
    test: adds children to Pareto store
    shrinkspace(Pareto store): reduce search space
End While
```

Fig. 2. Main algorithm of our PAS^4

```
for each child ch ∈ children {
    if (PS_i ∈ Pareto store ⪯ ch) then next ch
    if (ch ⪯ PS_i ∈ Pareto store then
        delete all PS_i
        add ch
}
```

Fig. 3. Pseudo-code of **test(Pareto store, λ children)** (called by **main**).

PAS^4'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 progresses. Eventually, upon termination, the size of the search space being inspected will be very small and will contain the solution desired (this is in the case of single-objective problems. For multi-objective problems, the final inspected zone will contain the feasible region). The main algorithm of PAS^4 is shown in Figure 2. Its goal is the construction of the Pareto front which is stored in an external memory (called *Pareto store*). The algorithm performs *MaxGen* loops. In each loop, as many as μ parents are picked to populate a $(\mu + \lambda)$-ES algorithm which runs for a certain number of generations. When the *Pareto store* cannot provide μ parents, the available parents are mutated in order to provide the required individuals. The λ children are inserted into the *Pareto store* using the procedure *test* and their positions on the grid are recorded. Then, the current Pareto set (in the *Pareto store*) is passed to the procedure *shrinkspace*, where new boundaries are determined for each decision variable. The grid is recomputed, as well as the new position of the individuals.

Most of **test(Pareto store, λ children)** is devoted to two things: (1) decide whether a child should be inserted in *Pareto store*, and if so, (2) how to make room for the new member. Also, after some iterations of the evolution strategy, (controlled by *maxffeval*), the space is shrunk around the current Pareto front represented by the individuals in *Pareto store*. The pseudo-code of this function is depicted in Figure 3.

A. Shrinking the Objective Space

The function **shrinkspace(Pareto store)** contains the most important contribution of this paper since its task is the selection of individuals, and the reduction of the search space. The pseudo-code of **shrinkspace(Pareto store)** is shown in Figure 4.

```
x_pob: smallest value of each x_i ∈ X
x̄_pob: largest value of each x_i ∈ X
select(file);
getMinMax( file, x_pob, x̄_pob);
trim(x_pob, x̄_pob );
adjustparameters(file);
```

Fig. 4. Pseudo-code of **shrinkspace(file)** (called by **main** of PAS^4)

```
m: number of constraints
i: constraint index
maxsize: max size of Pareto store
minPareto: 15% of Pareto store
constraintvalue(x,i):
    value of individual at constraint i
sortfile(Pareto store):
    sort by objective function
worst(Pareto store,i):
    worst individual in Pareto store for constraint i
validconstraints={1,2,3,...,m};
i=firstin(validconstraints);
While (size(Pareto store) > minPareto
and size(validconstraints) > 0) {
    x=worst(Pareto store,i)
    if (x violates constraint i)
        Pareto store=delete(Pareto store,x)
    else
    validconstraints=removeindex(validconstraints,i)
if (size(validconstraints) > 0) i=nextin(validconstraints)
}
if (size(Pareto store) == minPareto))
    list=Pareto store
else
    file=sort(Pareto store)
    selected=copy(Pareto store,minPareto)
```

Fig. 5. Pseudo-code of **select(Pareto store)** (called by **shrinkspace**)

In the following, we describe the four tasks performed by **shrinkspace**:

1) The function **select(Pareto store)** selects the best 15% individuals found in *Pareto store*, meeting the requirements listed in Section V. The selection algorithm is shown in Figure 5. Note that $validconstraints$ (a list of indexes to the problem constraints) indicates the ordering in which constraints are processed. The loop steps over the constraints removing only one (the worst) individual for each constraint. At least 15% of Pareto set must stay in *Pareto store*. When no more elements can be deleted (because they are feasible), the best 15% from *Pareto store* are chosen.
2) The function **getMinMax(Pareto store)** takes the chosen individuals in *selected* (last step in Figure 5) and finds the extreme values of each decision variable represented in *selected*. Thus, the vectors $\underline{\boldsymbol{x}}_{pob}$ and $\overline{\boldsymbol{x}}_{pob}$ are found.
3) 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**($\underline{\boldsymbol{x}}_{pob}$, $\overline{\boldsymbol{x}}_{pob}$) (see Figure 6) determines the new boundaries for the decision variables.
   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 H is a fraction α of its previous

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

Fig. 6. Pseudo-code of **trim** (called by **shrinkspace**)

value. The function **trim** first finds in the population the boundary values of each decision variable: $\overline{x}_{pob,i}$ and $\underline{x}_{pob,i}$. Then the new vectors $\overline{x}_i$ and $\underline{x}_i$ are updated by $deltaMin_i$, which is the decrement in each variable that in the overall reflects a change in the volume by a factor β. In PAS^4, all objective variables are reduced at the same rate β, therefore, β can be deduced from α as discussed next. Since we need the new hypervolume to be a fraction α of the previous one,

$$H_{\text{new}} \geq \alpha H_{\text{old}} \tag{8}$$

$$\prod_{i=1}^{n}(\overline{x}_i^{t+1} - \underline{x}_i^{t+1}) = \alpha \prod_{i=1}^{n}(\overline{x}_i^{t} - \underline{x}_i^{t})$$

Either x_i is reduced at the same rate β, thus

$$\prod_{i=1}^{n}\beta(\overline{x}_i^{t} - \underline{x}_i^{t}) = \alpha \prod_{i=1}^{n}(\overline{x}_i^{t} - \underline{x}_i^{t})$$
$$\beta = \alpha^{\frac{1}{n}}$$

In short, the new search interval of each decision variable x_i is adjusted as follows (the complete algorithm is shown in Figure 4):

$$width_{new} \geq \beta \times width_{old}$$

At this point of the description, an obvious question is: what is the ideal number of generations that we should run our ES? Also, what is a good space reduction ratio? For the sake of space, we will only indicate that we have thoroughly studied this issue, and found a somewhat unexpected result. We found that there are not as many combinations to play with as one might think at first sight. By setting the variable α to some value in the range [90, 95]%, and calling **shrinkspace()** every two or three generations of the ES algorithm (right hand side loop in Figure 1), PAS^4 exhibits a very good performance. In fact, in our experiments, we set $\alpha = 90\%$ and called shrinkspace() every 2 generations, and tried to improve the behavior by changing the population size. Thus, we claim that these parameters can remain fixed *for any problem.*

The variable *slack* is calculated once every new search interval is determined (usually set to 5% of the interval). The role of *slack* is simply to prevent (up to some extent) fast decreasing rates of the search interval.

4) The last step of **shrinkspace()** is a call to **adjustparameters(file)**. The goal here is to re-initialize the control variable σ through:

$$\sigma_i = (\overline{x}_i - \underline{x}_i)/\sqrt{n} \quad i \in (1, \ldots, n) \tag{9}$$

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].

Elitism

A special form of elitism is implemented by PAS^4 to prevent the lost of the best individual. Elitism is implemented as follows: the best individual a is stored and only replaced by the best b of any generation if b is better than a accordingly to the selection rules (see Section V).

PAS^4 for Optimizing problems in Discrete Search Spaces

Simple modifications are required for discrete optimization problems. In such case, the initial value of all the decision variables will be a random integer drawn from an uniform distribution, and bounded by the upper and lower limits indicated by the specific problem. Mutation of the decision variables is performed in this case as follows,

$$x_i^{t+1} = x_i^t + rand(\sigma_i)$$

where σ_i is the control variable of the corresponding decision variable, and $rand(\sigma_i)$ is a random number with uniform distribution in the interval $[0, \sigma]$. Control variables σ_i are mutated as follows,

$$if(random() < 0.45) \;\; then\; \sigma = \sigma + 1; \;\; else\; \sigma = \sigma - 1;$$

this is, with little less probability than the average of 0.5, the control variables diminish their value by 1. The reduction of the search space is performed as shown in Figure 6 for the real numbers space case, except that all results of the computations must be rounded up to the next integer. The variable $slack$ is also computed as depicted in Figure 6. Its value must also be rounded up, and its smallest possible value is 1.

VII. Examples

The parameters for all experiments described in Sections VII-B, VII-C, and VII-D, are: (1+1)-ES, Pareto store capacity for 200 individuals, 50% the minimum size of Pareto store, volume preserved $\alpha \geq 90\%$, call shrinkspace() every 2 generations, for 500 generations.

A. Solving Michalewicz's benchmark

Our first example is the solution of the well-known Michalewicz's benchmark for constrained optimization [12]. This benchmark, later extended by Runarsson & Yao [15], contains a set of 13 single-objective problems with constraints of different types (linear, nonlinear, equality and inequality). All these test problems have continuous search spaces and present different dimensionalities. In the past, a number of evolutionary algorithms with special constraint-handling schemes have been validated using this benchmark [10], [15], [7]. However, we will present a comparison of results only with respect to the Stochastic Ranking approach proposed by Runarsson & Yao [15], since this algorithm is representative of the state-of-the-art in the area and has been used recently as a reference to validate new constraint-handling techniques. In order to allow a fair comparison, we set the total number of fitness function evaluations to 350,000, as in [15]. *Pareto_store* can hold up to 200 individuals, which means that we adopt a (150+200)-ES. We also reduce at most 10% of search hypervolume at every 2 generations. We apply discrete crossover on the decision variables, and intermediate crossover on the control variables. Table I shows a comparison of the results obtained by PAS^4 with respect to Stochastic Ranking.

B. Optimization of a 49-bar Plane Truss

The next example chosen is the optimization of the 49-bar plane truss shown in Figure 7. The solutions to this problem were calculated in discrete search space using the catalog of *Altos Hornos de México*. We will describe next both a single-objective and multi-objective version of the problem.

1) The 49-bar Plane Truss as a Single-Objective Optimization Problem with Constraints: In this case, 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 $F(\vec{x}) = \sum_{j=1}^{49} \gamma A_j L_j$, 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 the catalog of *Altos Hornos de México, S.A.*, with 65 entries for the cross-sectional areas available for the design. Other relevant information is the following: Young modulus $= 2.1 \times 10^6$ kg/cm^2, maximum allowable stress = 3500.00 kg/cm^2, $\gamma = 7.4250 \times 10^{-3}$ kg/cm^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 three cases:

1) **Case 1. Stress constraints only:** Maximum allowable stress = 3500.00 kg/cm^2. A total of 49 constraints, thus 50 objective functions.
2) **Case 2. Stress and displacement constraints:** Maximum allowable stress = 3500.00 kg/cm^2, maximum displacement per node $= 10$ cm. There are 72 constraints, thus 73 objective functions.
3) **Case 3. 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.

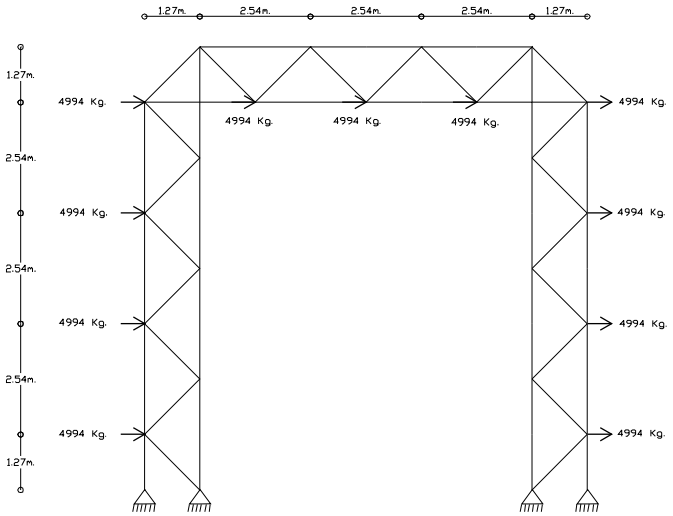

Fig. 7. 49-bar plane truss of the second optimization example.

TABLE II
COMPARISONS FOR THE 49-BAR TRUSS, CASE 1.

Algorithm	Average Weight (Kg)
PAS^4	610
SA	627
GA50	649
GSSA50	619
GSSA5	625

The average result of 30 runs for each case are shown in Tables II, III and IV. We compare PAS^4 with previous results (based on the catalog of Altos Hornos de México) reported by Botello et al. [2] using other heuristics with a penalty function [14] (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 clearly see that except for the GSSA50 technique, PAS^4 produced the lowest average weight.

2) The 49-bar Plane Truss as a Multi-Objective Optimization Problem with Constraints: The statement of this problem is similar to case 3 in Section VII-B.1, but now we consider two objective functions for simultaneous optimization. Our first objective is the minimization of the structure's weight; the second objective is the minimization of the horizontal displacement of the node at upper right corner of the structure. The

TABLE III
COMPARISONS FOR THE 49-BAR TRUSS, CASE 2.

Algorithm	Average Weight (Kg)
PAS^4	725
SA	737
GA50	817
GSSA50	748
GSSA5	769

TABLE I

COMPARISON OF PAS^4 WITH RESPECT TO STOCHASTIC RANKING (SR) [15].

		Best Result		Mean Result		Median Result	
Problem	**Optimal**	PAS^4	**SR**	PAS^4	**SR**	PAS^4	**SR**
g01	-15.0000	-14.9998	-15.0000	-14.88731	-15.000	-14.99645	-15.000
g02	0.803619	0.80346	0.803515	0.79901	0.781975	0.80330	0.78580
g03	1.000000	1.00017	1.000	1.00038	1.000	1.00039	1.00000
g04	-30665.5390	-30665.5300	-30665.5390	-30665.5300	-30665.539	-30665.5300	-30665.539
g05	5126.49800	5126.5200	5126.497	5180.15545	5128.881	5152.8950	5127.372
g06	-6961.8140	-6961.8100	-6961.814	-6961.8100	-6875.940	-6961.8100	-6961.814
g07	24.30621	24.33060	24.307	24.57961	24.374	24.46830	24.357
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630	680.630	680.630	680.63243	680.656	680.632	680.641
g10	7049.331	7059.840	7054.316	7366.9965	7559.192	7342.6000	7372.613
g11	0.750000	0.7500	0.7500	0.74993	0.750	0.74991	0.750
g12	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
g13	0.053950	0.053950	0.053957	0.22022	0.057006	0.08968	0.057006

TABLE IV

COMPARISONS FOR THE 49-BAR TRUSS, CASE 3.

Algorithm	Average Weight (Kg)
PAS^4	2603
SA	2724
GA50	2784
GSSA50	2570
GSSA5	2716

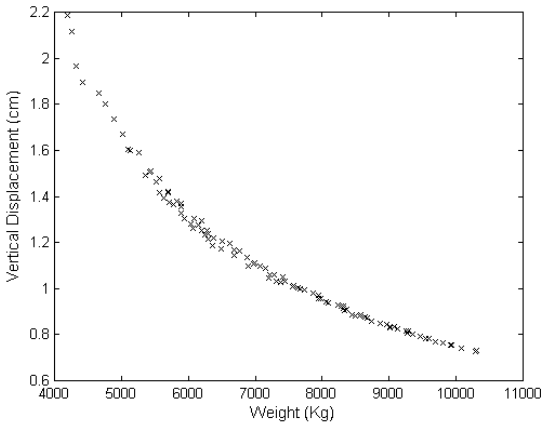

Fig. 8. Pareto front for the 49-bar plane truss of Figure 7 stated as a bi-objective optimization problem (See Section VII-B.2).

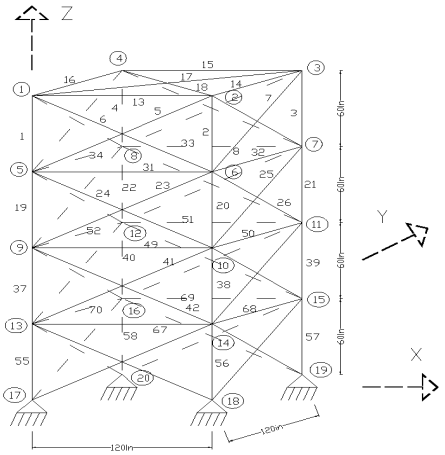

Fig. 9. Optimization of a 72-bar 3D structure.

TABLE V

THE 72-BAR 3D TRUSS CROSS SECTIONS BY GROUP.

Group	**Member**	**Group**	**Member**
1	A_1-A_4	9	A_{37}-A_{40}
2	A_5-A_{12}	10	A_{41}-A_{48}
3	A_{13}-A_{16}	11	A_{49}-A_{52}
4	A_{17}-A_{18}	12	A_{53}-A_{54}
5	A_{19}-A_{22}	13	A_{55}-A_{58}
6	A_{23}-A_{30}	14	A_{59}-A_{66}
7	A_{31}-A_{34}	15	A_{67}-A_{70}
8	A_{35}-A_{36}	16	A_{71}-A_{72}

Pareto front of these two objectives subject to 71 constraints is shown in Figure 8.

C. Optimization of 72-bar 3D Structure

The following problem is the design of the 72-bar 3D structure shown in Figure 9. The truss is subject to two distinct loading conditions and sixteen independent design variables. All nodes are subject to a displacement constraint $\Delta \leq 0.25$ inches in x and y direction. All bars have a stress constraint -1759.25 kg/cm^2 $\leq (\sigma_a)_i \leq 1759.25$ kg/cm^2, $i = 1, 2 \ldots 72$. The minimum size constraint is 0.254 cm^2 $\leq A_i, i = 1, 2 \ldots 72$. The properties of the material are: modulus of elasticity: 7.031×10^6kg/cm^2, volumetric weight: 2.77×10^{-3} kg/cm^3. The first loading condition has a point load in node 1 with 2270 kg in x direction, 2270 kg in y direction and -2270 kg in z direction. The second loading condition has four load points in nodes 1,2,3 and 4, with -2270 kg in z direction. The design problem is the design of the truss for both loading conditions. In Table V, we provide the group description of the truss. We solved this problem as a single-objective optimization case in both continuous and discrete search spaces.

1) The 72-bar 3D Structure in Continuous Search Space as a Single-Objective Optimization Problem with Constraints: As noted, the design problem is the minimization of the weight structure subject to both loading conditions. We compared

TABLE VI

PAS^4 VS. RESULTS OF SEVERAL AUTHORS FOR OUR 72-BAR 3D STRUCTURE IN CONTINUOUS SEARCH SPACE.

Algorithm	Best Minimun Weight (Kg)
PAS^4	172.02
Venkayya [18]	173.06
Gellatly [6]	179.77
Renwei [13]	172.36
Schmit[16]	176.44
Xicheng [19]	172.90
GAOS [2]	173.94

TABLE VII

PAS^4 STATISTICS FOR THE 72-BAR 3D STRUCTURE IN CONTINUOUS SEARCH SPACE.

Parameter	Weight (Kg)
Best	172.02
Worst	172.09
Mean	172.05
Std. dev.	0.015
Median	172.04
Fact. Sol.	30

PAS^4 against several results of other authors in Table VI; as it can be observed our approach provided the best solution. In Table VII, we show basic statistics for 30 runs.

2) The 72-bar 3D Structure in Discrete Search Space as a Single-Objective Optimization Problem with Constraints: We solved three cases of this problem using the catalog of *Altos Hornos de México, S.A.* with 65 entries for the cross-sectional areas: 1) stress constraints only; 2) stress and displacement constraints; 3) displacement constraints, and considers bar traction and compression stress, as well as their proper weight. The values of material properties and constraints remain with no change for all three cases. Statistics of the best solutions in 30 runs for the 3 cases are shown in Table VIII.

D. Optimization of a Steel Dome

Our last example is the optimization of the weight of the steel dome shown in Figure 10 for discrete search space (using the catalog of *Altos Hornos de México, S.A.*)

1) The Steel Dome as a Single-Objective Optimization problem with constraints: This truss has seven independent design variables. For all nodes, a displacement constraint $\Delta \leq 20$ mm was assumed in the z direction. All bars are subject to the stress constraint given by the aforementioned catalog. The

TABLE VIII

PAS^4 SOLUTIONS TO THE 72-BAR 3D STRUCTURE USING A CATALOG WITH 65 ENTRIES (DISCRETE SEARCH SPACE).

Parameter	Case1 (Kg)	Case2 (Kg)	Case3 (Kg)
Best	92.3295	192.7194	630.400
Worst	92.3295	193.4353	640.3640
Mean	92.3295	192.9098	633.2354
Std. dev.	0.0	0.3060	2.7371
Median	92.3295	192.7194	632.9665
Fact. Sol.	30	30	30

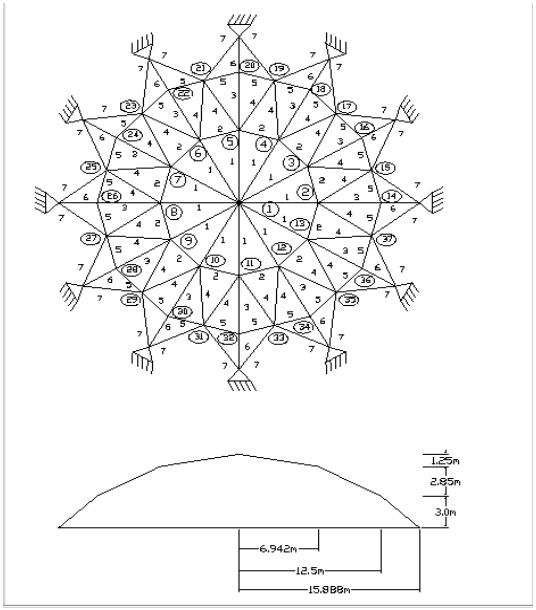

Fig. 10. Optimization of a steel dome.

TABLE IX

PAS^4 SOLUTIONS TO THE STEEL DOME (DISCRETE CASE).

Parameter	Case1 (Kg)	Case2 (Kg)	Case3 (Kg)
Best	703.57	703.57	13642.33
Worst	703.57	703.57	13651.93
Mean	703.57	703.57	13644.56
Std. dev.	0.0	0.0	4.1304
Median	703.57	703.57	13642.33
Fact. Sol.	30	30	30

material properties are: modulus of elasticity= $2.1\text{x}10^6$kg/cm^2, yield strength: 2750 kg/cm^2. The loading conditions are a punctual load in z direction with different magnitude: -500 kg in node 1; -40 kg in nodes 17, 23, 29, and 35; -120 kg in nodes 16, 18, 22, 24, 28, 30, 34, and 36; -200 kg on the rest of the nodes. We solved three cases of this problem using the catalog of *Altos Hornos de México, S.A.* with 65 entries for the cross-sectional areas: 1) stress constraints only; 2) stress and displacement constraints; 3) displacement constraints, and considers bar traction and compression stress, as well as their proper weight. The values of material properties and constraints remain with no change for all three cases. Solutions for the 3 cases are shown in Table IX. Columns "Case1" and "Case2" are equal because displacement constraints are too small.

2) The Steel Dome as a Multi-Objective Optimization Problem with Constraints: In this case, the two objective functions are the minimization of the weight structure, and the vertical deformation of central node, subject to the original constraints. In Figure 11 we show the Pareto front for this problem. The plot shows that there are many weight combinations that keep the vertical deformation practically with no change. It can also be seen that this deformation increases, perhaps dangerously, with very little variations of the weight, when this is in the

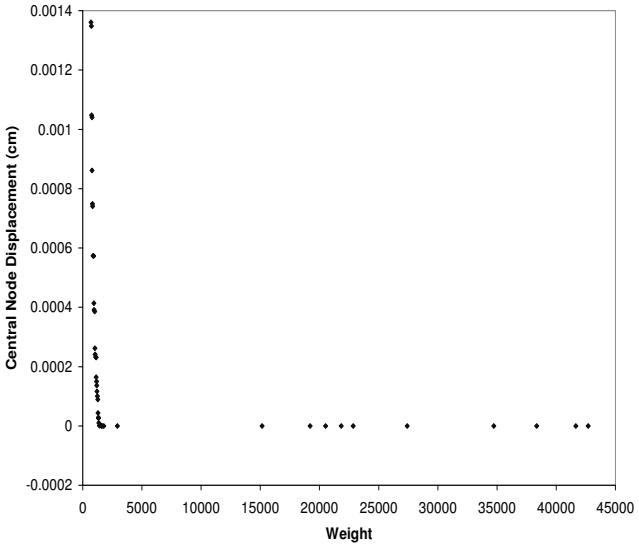

Fig. 11. Pareto front of the steel dome stated as a bi-objective problem.

critical interval $[1000, 2000]$ (approximately).

VIII. Discussion and Final Remarks

We have introduced the PAS^4 evolutionary algorithm that combines the following three ideas: 1) a constraint-handling mechanism based on multi-objective optimization concepts; 2) a Pareto dominance-based selection operator which promotes diversity and a desired blend including promissory and "best unfeasible" individuals; 3) a search reduction population-driven mechanism that directs the search towards promising areas of the space; and 4) an external memory to store the latest nondominated solutions found. Regarding constraint-handling for single-objective optimization, we compared our results with respect to stochastic ranking [15] (see Table I). We can see that our results are highly competitive. Furthermore, we have shown the potential of our proposal by applying it to problems both in continuous and in discrete search spaces. Pareto fronts in these spaces have been plotted from our experiments. PAS^4 requires to make decisions over four parameters (described in Section VII): the size of the Pareto set; the shrinkspace rate; the size of the new hypervolume after reduction; and the percentage of remanent "best infeasible" individuals in the Pareto set. These parameters are not hard to set, and we argue that their fine-tuning is not more difficult than that required by the traditional parameters of a genetic algorithm. Our approach is not the simplest possible algorithm for evolutionary optimization, but it has as an aggregated value the fact of being able to deal (rather competitively) with both constrained single-objective and multi-objective optimization problems.

Acknowledgments

The first and second authors acknowledge support from CONACyT through projects P40721-Y and 42523. The third author acknowledges support from CONACyT through project number 42435-Y.

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] 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.

[4] 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.

[5] Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.

[6] R. A. Gellatly and L. Berke. Optimal Structural Design. Technical Report AFFDL-TR-70-165, Air Force Flight Dynamics Laboratory, 1971.

[7] Sana Ben Hamida and Marc Schoenauer. ASCHEA: New Results Using Adaptive Segregational Constraint Handling. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 884–889, Piscataway, New Jersey, May 2002. IEEE Service Center.

[8] Arturo Hernández Aguirre, S. Botello, G. Lizárraga, and C. Coello. ISPAES: A Constraint-Handling Technique Based on Multiobjective Optimization Concepts. In *Proceedings of the 2nd International Conference, Evolutionary Multi-Criterion Optimization (EMO'2003)*, pages 73–87, Berlin, Germany, April 2003. Springer-Verlag, Lecture Notes in Computer Science No. 2632.

[9] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[10] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[11] Efrén Mezura-Montes and Carloa A. Coello Coello. 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, pages 6–13, Piscataway, New Jersey, December 2003. Canberra, Australia, IEEE Service Center.

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

[13] X. Renwei and L. Peng. Structural optimization based on second order approximations of functions and dual theory. *Computer Methods in Applied Mechanics and Engineering*, 65:101–104, 1987.

[14] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some Guidelines for Genetic Algorithms with Penalty Functions. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197. Morgan Kaufmann Publishers, 1989.

[15] T.P. Runarsson and X. Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[16] L.A. Schmidt and B. Farshi. Design of Optimum Structures. *Journal of the American Institute of Aeronautics and Astronautics*, 12:231–233, 1974.

[17] Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3):391–412, 1997.

[18] VB. Venkayya. Design of Optimum Structures. *Computers & Structures*, 1:265–309, 1971.

[19] W. Xicheng and M. Guixu. A Parallel Iterative Algorithm for Structural Optimization. *Computer Methods in Applied Mechanics and Engineering*, 96:25–32, 1992.