IS-PAES: MULTIOBJECTIVE OPTIMIZATION WITH EFFICIENT CONSTRAINT HANDLING

Arturo Hernández Aguirre
Salvador Botello Rionda
Giovanni Lizárraga Lizárraga
Center for Research in Mathematics
Department of Computer Science, Guanajuato, Gto. 36240, MEXICO
artha,botello,giovanni@cimat.mx

Carlos Coello Coello
CINVESTAV-IPN, EE Dept., Computer Science Section
México, D.F. 07300, MEXICO
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, IS-PAES does not have the serious scalability problems of the PAES. The viability of the proposed approach is validated with several examples taken from the standard evolutionary and engineering optimization literature.

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, 7]. 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 [9, 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) [6]. Our approach can be use 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 briefly describes the work related to our own. In Section 3, we describe the main algorithm of IS-PAES. Section 4 provides a comparison of results and Section 5 draws our conclusions and provides some paths of future research.

2. 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 $f(\vec{x})$ 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 [5] 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 $f_i(\vec{x})$=0 for $1 \leq i \leq m$ and $f(\vec{x}) \leq f(\vec{y})$ for all feasible $\vec{y}$ (assuming minimization).

Based on this main idea, several approaches have proposed in the last few years. Some of them use population-based techniques (e.g., [3]), and others use Pareto ranking (e.g., [10]). However, all of these techniques are normally more useful to approach the feasible region, but are not as effective for reaching the global optimum of a problem. We argue in this paper that the main reason for having this limitation has to do with the focus of the search in traditional multiobjective optimization algorithms. Rather than focusing the effort on finding good "trade-offs"

(as in multiobjective optimization), we propose to focus the search in finding the boundary between the feasible and the infeasible regions and then concentrating the search effort on reaching the global optimum. Such is the nature of the algorithm proposed in this paper.

3. IS-PAES Algorithm

IS-PAES has been implemented as an extension of the Pareto Archived Evolution Strategy (PAES) proposed by Knowles and Corne [6] for multiobjective optimization. PAES 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 its 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 "shrinked" over time. Eventually, upon termination, the size of the search space being inspected will be very small and will contain the solution desired. The main algorithm of IS-PAES is shown in Figure 1.

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: $x_1 \square x_2$ means x_1 is located in a less populated region of the grid that x_2. The pseudo-code of this function is depicted in Figure 2.

3.1 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

```
maxsize: max size of file
c: current parent ∈ X (decision variable space)
h:child of c ∈ 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≤MaxNew do
    h = mutate(c); cnew+ =1;
    if (c≼h) then exit loop
    else if (h≼c) then { remove(c); add(g); c=h; }
    else if (∃ a_h ∈ file | a_h ≼ h) then exit loop
    else if (∃ a_d ∈ file | h ≼ a_d) then
        add( h ); ∀ 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
```

Figure 1. Main algorithm of IS-PAES

```
if (current < maxsize) then add(h)
    if (h □ c) then c = h
else if (∃a_p∈file | h □ a_p) then { remove(a_p); add(h) }
if (h □ c) then c = h;
```

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

fine, for in this case only IS-PAES is able to deal with any number of functions and granularity level.

3.2 Shrinking the objective space

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

$\underline{\boldsymbol{x}}_{pob}$: 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);

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

partially feasible, the best individuals are chosen from the *file* sorted by "objective-function" then by "constraint#1", then by "constraint#2", and so on. The function **getMinMax(file)** takes this list and finds the extreme values of the decision variables represented by those individuals. Thus, the vectors $\underline{\boldsymbol{x}}_{pob}$ and $\overline{\boldsymbol{x}}_{pob}$ are found. 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 $\in$ { 1, . . . , n }
$slack_i = 0.05 \times (\overline{x}_{pob,i} - \underline{x}_{pob,i})$
$width\_pob_i = \overline{x}_{pob,i} - \underline{x}_{pob,i}$; $width_i^t = \overline{x}_i^t - \underline{x}_i^t$
$deltaMin_i = \frac{\beta * width_i^t - width\_pob_i}{2}$
$\text{delta}_i = \max(\text{slack}_i, \text{deltaMin}_i)$;
$\overline{x}_i^{t+1} = \overline{x}_{pob,i} + delta_i$; $\underline{x}_i^{t+1} = \underline{x}_{pob,i} - delta_i$;
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** $\overline{x}_i^{t+1} = \overline{x}_{original,i}$;

Figure 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].

4. Comparison of Results

We have validated our approach with several problems used as a benchmark for evolutionary algorithms (see [7]) 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" which is representative of the state-of-the-art in constrained evolutionary optimization (see [8] for details). Regarding the engineering optimization problems, we compared results with respect to those previously reported in the literature.

4.1 Test problems used with EAs

The following parameters were adopted in this case: $maxsize = 200$, $bestindividuals = 15\%$, $slack = 0.05$, $r = 400$, $MaxNew = 350000$.

1 **Problem g06**

$$\text{Minimize } F(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{1}$$

subject to:

$$\begin{aligned} g_1(\boldsymbol{x}) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\ g_2(\boldsymbol{x}) &= (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \end{aligned} \tag{2}$$

where $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$. The global optimum is located at $\boldsymbol{x}^* = \{14.095, 0.84296\}$, with $F(\boldsymbol{x}^*) = -6961.81388$. Both constraint are active. The best solution found by IS-PAES is: $\boldsymbol{x} = \{14.0950000092\ 0.842960808844\ \}$, with $F(\boldsymbol{x}) = -6961.813854$.

Note in Table 1 how IS-PAES had a better average result and a lower standard deviation than R&Y.

Measure	IS-PAES	R&Y
Best	-6961.814	-6961.814
Worst	-6961.810	-6350.262
Average	-6961.813	-6875.940
Std. Deviation	8.5E-05	1.6E+02
Median	-6961.814	-6961.814
Feasible solutions	30	30

Table 1. Comparison of results for problem g06

2 **Problem g08**

$$\text{Minimize} F(\boldsymbol{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)} \tag{3}$$

subject to:

$$\begin{aligned} g_1(\boldsymbol{x}) &= x_1^2 - x_2 + 1 \leq 0 \\ g_2(\boldsymbol{x}) &= 1 - x_1 + (x_2 - 4)^2 \leq 0 \end{aligned} \tag{4}$$

where $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$. The global optimum is located at $\boldsymbol{x^*} = \{1.2279713, 4.2453733\}$, with $F(\boldsymbol{x^*}) = 0.095825$. The best solution found by IS-PAES was: $\boldsymbol{x}$ ={1.227971353, 4.245373368}, with $F(\boldsymbol{x}) = -0.095825041$. The performance of both algorithms is similar for this problem. Results are shown in Table 2.

Measure	IS-PAES	R&Y
Best	-0.095825	-0.095825
Worst	-0.095825	-0.095825
Average	-0.095825	-0.095825
Std. Deviation	0.0	2.6E-17
Median	-0.095825	-0.095825
Feasible solutions	30	30

Table 2. Comparison of results for problem g08

4.2 Optimization of a 49-bar plane truss

The last problem is the optimization of a 49-bar plane truss. 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 $F(\boldsymbol{x}) = \sum_{j=1}^{49} \gamma A_j L_j$, 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:
Case 1. Stress constraints only. Maximum allowable stress $= 3500.00 kg/cm^2$ A total of 49 constraints, thus 50 objective functions.
Case 2. 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.
Case 3. Real design problem. The design problem considers traction and compression stress on the bars, as well as the proper weight. Maximum allowable stress $= 3500.00 kg/cm^2$, maximum displacement per node $= 10cm$. A total of 72 constraints, thus 73 objective functions. The average result of 30 runs for each case are shown in Tables 3, 4 and 5. We compare IS-PAES with previous results reported by Botello [2] (SA: Simulated Annealing, GA50: Genetic Algorithm with a population of 50, and GSSA: General Stochastic Search Algorithm with populations of 50 and 5).

Algorithm	Average Weight (Kg)
IS-PAES	610
SA	627
GA50	649
GSSA50	619
GSSA5	625

Table 3. Comparison of different algorithms on the 49-bar struss, case 1

5. 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 in the area. As

Algorithm	Average Weight (Kg)
IS-PAES	725
SA	737
GA50	817
GSSA50	748
GSSA5	769

Table 4. Comparison of different algorithms on the 49-bar struss, case 2

Algorithm	Average Weight (Kg)
IS-PAES	2603
SA	2724
GA50	2784
GSSA50	2570
GSSA5	2716

Table 5. Comparison of different algorithms on the 49-bar struss, case 3

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, premature convergence may occur in some cases. 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 multiobjective evolutionary algorithms.

Acknowledgments

The first author acknowledges partial support from CONCyTEG project No. 01-02-202-111 and CONACyT No. I-39324-A. The second author acknowledges support from CONACyT project No. 34575-A. The last author acknowledges support from CONACyT project No. NSF-CONACyT 32999-A.

References

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

[2] Salvador Botello, José Luis Marroquín, and et al. Solving Structural Optimization problems with GAs and simulated annealing. *International Journal of Numerical Methods in Engineering*, 45:1–16, 1999.

[3] Carlos A. Coello Coello. Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization*, 32(3):275–308, 2000.

[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, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, June 2002. ISBN 0-3064-6762-3.

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

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

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

[9] Alice E. Smith and David W. Coit. Constraint Handling Techniques—Penalty Functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.

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