Multiobjective Shape Optimization using Estimation Distribution Algorithms and Correlated Information

Sergio Ivvan Valdez Peña, Salvador Botello Rionda, and
Arturo Hernández Aguirre

Center for Research in Mathematics (CIMAT)
Department of Computer Science
A.P. 402, Guanajuato, Gto. 36000, México
ivvan,botello,artha@cimat.mx

Abstract. We propose a new approach for multiobjective shape optimization based on the estimation of probability distributions. The algorithm improves search space exploration by capturing landscape information into the probability distribution of the population. Correlation among design variables is also used for the computation of probability distributions. The algorithm uses finite element method to evaluate objective functions and constraints. We provide several design problems and we show Pareto front examples. The design goals are: minimum weight and minimum nodal displacement, without holes or unconnected elements in the structure.

1 Introduction

Shape optimization has been widely tackled by evolutionary algorithms. Genetic algorithms, (GAs), have been applied to shape optimization problems with some success, providing feasible solutions with acceptable fitness value [1,3]. Nonetheless, GA based approaches present difficulties at finding solutions without holes or unconnected segments. This behavior can be explained by population diversity issues, which favor premature convergence and reduced search space exploration [2,4].

In this paper, we present a multiobjective algorithm for shape optimization (MASO), which is based on estimation distribution concepts. The approach uses binary representation, and makes calls to an external finite element system to evaluate the fitness of candidate structures (individuals). MASO is related to univariate marginal distribution algorithms (UMDA) [6], and to Population Based Incremental Learning (PBIL) [7]. Therefore, every g generations, MASO estimates a (biased) probability distribution by sampling the current Pareto set. The new random population is generated with the updated distribution.

We have improved the algorithm's performance by using specific knowledge derived from the problem domain. This information, combined with the current

Pareto set, provides better distribution estimations. Through experiments, we have observed enhanced exploration around promising areas, and less number of small holes and unconnected elements in the structure. Other approaches infer this relationship through the use of Bayesian probabilities [8,9].

2 Problem Definition

The problem is to find the set of structures which fulfill design constraints (stress), and optimizes: total structure weight and, node displacement in one or more nodes (see Figure 1). Also, a minimum number of "objects or pieces" in the structure is desired. Another desired characteristic for the resulting structure is a minimum number of "small holes".

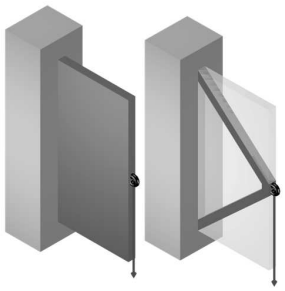


Fig. 1. Problem definition, initial search space and the minimum weight structure

In our problem, the design constraints are given by a maximum permissible Von Misses stress [10] (a standard criterion for mechanical design which represents the material resistance). The algorithm works on a delimited region (the whole piece) as it shown in Figure 1. The structure on the left side is the whole search region, that is, all structure's elements are present. The delivered design, shown on the right hand side, has minimum weight and minimum displacement (therefore, a member of the Pareto front). The design is achieved by "removing elements" from the structure, which is previously represented in discrete form for this purpose [11] (an "element" is one cell of the grid). Thus, binary representation is ad hoc for this problem. A "0" value represents a hole in the structure, while a "1" represents a given thickness value t. The discrete space and a representation example are shown in Figure 2.

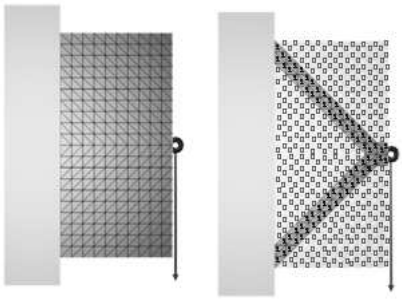


Fig. 2. Left: discrete search space; Right: representation of a structure configuration

3 Objective Functions and Constraints

As noted before, the design problem has two objective functions: the first objective is the minimization of the structure's weight, including the total number of "objects" needed to build the structure, and the number of "small holes". The second objective function accounts for displacement minimization at some nodes (user defined). An object is a set of at least two elements with one common side; likewise, a "small hole" is a non-present (0 value) element whose surrounded neighbors are present. Figure 3(a) shows a structure with 4 objects, and Figure 3(b) is a configuration example with 3 small holes.

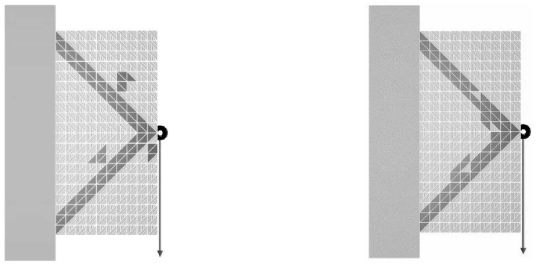


Fig. 3. (a) 4-Object configuration, (b) 3-small hole configuration

The first objective function is expressed as follows:

Minimize:

$$F(x, O_n, O_b) = [1 + c_1 * (O_n - 1) + c_2 O_b] W(x) \tag{1}$$

Where:

$$W(x) = \sum_{i=0}^{n} w_i x_i \tag{2}$$

In Equation 1, c_1 is the object penalization constant. This constant modifies the function if the number of objects is greater than 1, otherwise it is 0. On the other hand, the constant c_2 penalizes holes whose size is exactly one element. In all experiments reported in Section 6, we used $c_1 = c_2 = 10$. Variable n is the total number of elements (cells in the grid) in the structure; O_n is defined by the number of objects; O_b is the number of small holes. Equation 2 models the structure weight, where w_i represents each element weight, and x_i is the bit value at the $i - th$ position (present, not-present).

The second objective function, Equation 3, minimizes the displacement at some specific nodes.

Minimize:

$$G(\delta, O_n, O_b) = [1 + c_1(O_n - 1) + c_2 O_b] \sum_{j=0}^{m} |\delta_j| \tag{3}$$

Where $|\delta_j|$ is the absolute value of the displacement at the $j - th$ node. Finally, m is the number of nodes involved in the second objective function.

The design constraints, as we said before, represent the maximum Von Misses stress of each element (a standard mechanical criterion to evaluate the material resistance). Clearly, all elements must have a Von Misses stress value equal or lower than the maximum permissible for the material. Thus, in Equation 4, the sum of $\rho(\sigma)$ (first factor) represents the number of elements violating the Von Misses stress constraint. The second factor, represents the summation of the Von Misses stress of each element present in the structure.

$$H(x, \rho(\sigma), \gamma(\sigma)) = \left(\sum_{i=0}^{n} \rho_i(\sigma) \right) \left(\sum_{i=0}^{n} x_i \gamma_i(\sigma) \right) \tag{4}$$

Where:

$$\rho_i(\sigma) = \begin{cases} 0 & if\ (\sigma_M - \sigma_i) \geq 0 \\ 1 & if\ (\sigma_M - \sigma_i) < 0 \end{cases} \tag{5}$$

And:

$$\gamma_i(\sigma) = \begin{cases} 0 & if\ (\sigma_M - \sigma_i) \geq 0 \\ (\sigma_i - \sigma_M) & if\ (\sigma_M - \sigma_i) < 0 \end{cases} \tag{6}$$

Where σ_M is the maximum permissible Von Misses stress, and σ_i is the Von Misses stress of each element.

4 Implementation

The probability vectors can be initialized in three different ways: a) with random numbers in the [0,1] interval, b) with all vector probability values equal to 0.5, and c) according to the algorithm in Section 4.1. The last approach was used for the experiments of this paper. The population is generated as described in Section 4.2, which is a common procedure used by EDA's with binary representation [7]. The objective and constraint functions are evaluated with triangular finite element standard routines [11]; the selection mechanism is Pareto dominance taking the constraint as one more objective function. The main components of MASO are shown in Figure 4. In order to update the probability vectors, the current Pareto set is sorted by weight (see Section 4.3).

When the probability vectors have lost their exploration capacity (the variance is smaller than a threshold value, see Section 4.5), a membrane filter is applied to them in order to smooth the probability. The pseudo code of MASO algorithm is shown in Figure 5.

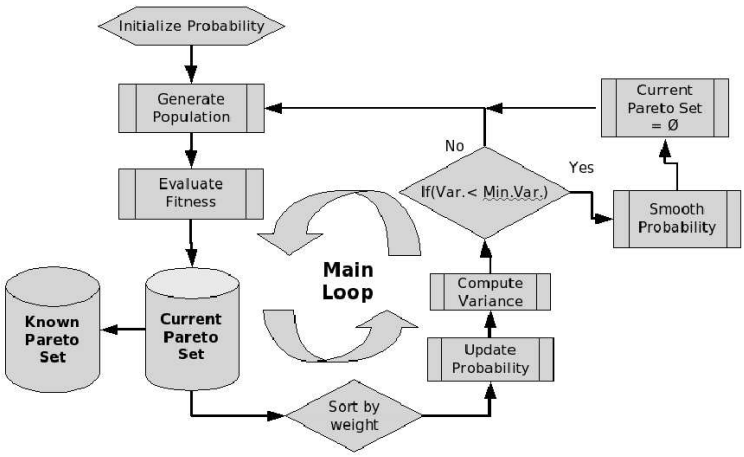


Fig. 4. Main loop of MASO depicting principal routines

4.1 Probability Initialization

In Figure 6 we show an initialization procedure for probability vectors. Basically, we compute the stress values for "all-1" individuals, then we set to 0 the bits which represents the elements with low Von Misses stress value. A preset

```
Enter the next parameters:
μ: Smoothing Parameter. The weight of the gradient penalization in the membrane
filter (see [5] and Equation 9)
MinVar: Minimum Variance criterion to apply the membrane filter.
LSP,LIP: Superior and Inferior Probability Limit.
NV: Number of Probability Vectors.
NIM: Number of Individuals Generated by Every Probability Vector.
λ: parameter for the probability updating (learning rate,see subsection 4.3 ).
NG: Number of generations.

Probability_Initialization ();
k < − 0
i < − 0
while (i < NG ) {
  Generate_Population ();
  Evaluate_Objective_and_Constraints_Functions ();
  Select_Non-Dominated_Individuals ();
  Sort_Individualsbyweight();
  Update_Probability_Distributions ();
  Compute_Variance ();
  If (ComputedVariance < MinVar) {
    Smooth_Probability_Distributions ();
    Update_External_File ();
    k++; }
  i++;  } end
```

Fig. 5. Pseudo-code of Multiobjective Algorithm for Shape Optimization

threshold is used in this decision. P and O are structure configurations; O in particular is the configuration with all bit values equal to 1, $VonMissesStress_i$ is the stress in the $i-th$ element, $InfProbLim$ and $SupProbLim$ are probability limits, and the function $Solve\_VonMisses(Configuration)$, solves the FE problem for a particular configuration.

4.2 Generation of the population

The population is generated by a set of k distribution probability vectors called V_l . Every vector V_l generates q structure configurations called I_j. Thus, every bit from I_j is generated by a Bernoulli experiment with a success probability p_i.

4.3 Updating probability distributions

The non-dominated individuals are used to compute and update the probability distributions. First, we find the non-dominated structures by treating the stress constraint as an additional objective. Thus, dominance is computed with three

```
  for i=1..NoElements O[i]=1 endfor
  Solve_VonMisses(O);
  Interval = (maxVonMisses-minVonMisses)/(NoElements)
  threshold = maxVonMisses - Interval
Step 1:
  For i=1..NoElements
     if (VonMissesStress_i < threshold)   P[i] = 0
    Else    P[i] = 1
   endfor
  if ( IsNotFeasible(P) ) {# True when is infeasible
   threshold = threshold - Interval
   go to: Step 1 }
  Interval = (threshold - minVonMisses )/( NoVectors-1 );
  For i= 0..NoVectors {
     For k=0..NoElements {
        if ( VonMissesStress_i < threshold)
          ProbabilityVector_k,i = InfProbLim;
        else
          ProbabilityVector_k,i = SupProbLim; }
     threshold = threshold - Interval; }
```

Fig. 6. Pseudo-code for probability initialization

functions (two objectives plus the constraint). In this non-dominated temporal set there are feasible and infeasible individuals; the infeasible ones are sorted by total amount of constraint violation. All feasible individuals plus a small percentage of the unfeasible ones (those with a small constraint value) are used to update the probability distributions. For our experiments we used 10% of the infeasible structures. Before updating the distributions, all structures are sorted by one of the objectives; in our case we use the weight value to sort the structures.
Thus, being C_f the number of feasible non-dominated structures, and C_u the number of non-feasible non-dominated structures, ζ represents the percentage of non-feasible structures that are taken to update the probability vectors.

$$C_t = C_f + \zeta C_u \tag{7}$$

For k probability vectors, the new probability vector will be:

$For\ j = 1..Number\ of\ Bits$

$$V_{l,j}^{t+1} = \lambda V_{l,j}^{t} + (1-\lambda) \sum_{i=(l-1)*Round(C_t/k)}^{(l)*Round(C_t/k)} I_{i,j}^{b} / Round(C_t/k) \tag{8}$$

Where $V_l^{(t+1)}$ is the new probability vector for generation $t+1$, V_l^t is the probability vector at generation t, λ is a memory factor(learning rate, see [7]) that preserves the knowledge of the last distribution, I_i^b are the binary arrays (non-dominated individuals) that will be used to update the vector V_l. The $Round()$ function returns the nearest integer from the real result of C_t/k. As we can see, $\sum_{i=(l-1)*Round(C_t/k)}^{(l)*Round(C_t/k)} I_i^b/Round(C_t/k)$ is nothing else than the mean of a set of binary arrays. Note that the selection of non-dominated individuals is performed over the current population and the current Pareto set.

4.4 Smoothing the probability distributions

Due to the implicit reinforcement of the non-dominated individuals, Equation 8 takes the probability vectors to values close to "0", or to "1" . When this happens, the probability vectors have lost their exploration capacity. For MASO, the probability distribution vectors are smoothed, so predominant peaks are removed from the probability surface. Restarting the population with new probability distributions enhances exploration and local minima avoidance.

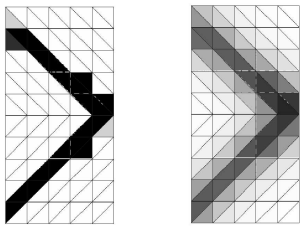
Probability distributions are smoothed by a membrane filter [5]. Equation 9 finds a smooth function f which preserves the shape of the original distribution g, but will avoid the abrupt changes. The abrupt changes are penalized by the gradient ∇f.

$$U(f) = \int_{\omega} ([f(x,y) - g(x,y)]^2 + \mu||\nabla f(x,y)||^2)d\omega \tag{9}$$

Where $f(x,y)$ is the function which minimize the functional $U(f)$, $g(x,y)$ is the original probability distribution function, μ is a parameter which regulates how smooth f will be. In our experiments we used a μ value in the interval $0.35-0.5$. In this case, we approximate the gradient with the difference between the central element and each neighbor (we have three neighbors for a triangular mesh).

Figure 7 (a) shows a probability distribution (one vector V_l^t) before applying the membrane filter. Probabilities values are shown in gray scale color, where values close to 1 are shown in black, and probabilities close to 0 in white. In that figure we can observe that the probability distribution can generate a good approximated solution but, if we do not modify the probabilities they are not able of keeping search space exploration. In Figure 7 (b) we can observe how the membrane filter has spread the probability distribution over the neighbors.

We identify the poor search capacity of a probability vector by a variance measure. This measure is computed as explained in Section 4.5.
Once the smoothing process is finished, we must update the set of non-dominated individuals stored in the external file, (named *Known Pareto Set* in Figure 4). The new population is generated anew using the smoothed probability distributions; eventually the population will converge, the distributions smoothed again,



(a) **(b)**

Fig. 7. Relationship between a structure and its probability distribution: (a) before applying the membrane filter, (b) after applying the membrane filter.

and another *Current Pareto Set* will be generated and used to update the external file.

4.5 Variance computation

We measure the variance of those locations whose probability vector are in the interval $[0.0002, 0.9998]$. The membrane filter is applied to every probability distribution if the computed variance is smaller than a threshold. Pseudo code of variance computation algorithm is shown in Figure 8.

```
ComputedVariance=0
For l=1.. NoVectors {
   Variance=0
   For=1..NoElementos {
    if (ProbabilityVector_{l,i} ≤ InfProbLim && ProbabilityVector_{l,i} ≥ SupProb-
Lim)
      Variance=Variance+1; }
  if (Variance > ComputedVariance)
   ComputedVariance =Variance }
```

Fig. 8. Pseudo-code of variance computation algorithm

5 Metrics

A MOEA convergence metric proposed by Deb and Jain is computed to measure the convergence and behavior of the algorithm [12].

5.1 Convergence metric

A reference set P^* is determined from the union of 30 Pareto sets (therefore, Known Pareto set of each run). This metric takes a normalized value within $[0, 1]$; near 0 means better [12]. The convergence metric is computed as follows:

1. Identify the no dominated set $F^{(t)}$ of $P^{(t)}$ (a population)
2. Form each point i in $F^{(t)}$, calculate the smallest normalized Euclidean distance to P^* as in Equation 10, f_k^{max} and f_k^{max} are the maximum and the minimum function values of k-th objective function in P^*

$$d_i \; {}^{|P^*|}min_{j=1} \sqrt{sum_{k=1}^{M} \left(\frac{f_k(i) - f_k(j)}{f_k^{max} - f_k^{min}} \right)} \tag{10}$$

3. Calculate the convergence metric by averaging the normalized distance for all points in $F^{(t)}$:

$$C(P^{(t)}) = \frac{sum_{i=1}^{|F^{(t)}|} d_i}{|F^{(t)}|} \tag{11}$$

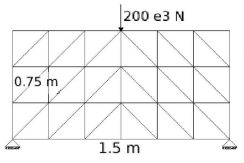
Deb and Jain proposed to normalize the convergence metric by the maximum value (usually $C(P^{(0)})$) : $\hat{C}(P^{(t)}) = C(P^{(t)})/C(P^{(0)})$).To compute the convergence metric we calculate the normalized distance for the vectors in F^*, and if $|P^*| > |F^*|$ (we have more points in P^* than F^*) we set $|P^*| - |F^*|$ distances equal to 1.

6 Experiments

Two study cases are used to test our approach. In the first case we know the true Pareto set, whilst in the other it is unknown.

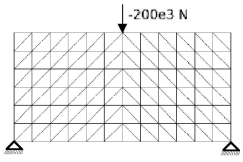
6.1 Experiment 1 (description)

A simple structure with a punctual load is supported at the lower corners. The dimensions and load magnitude are shown in Figure 9. We chose this problem because the true Pareto set is easy to compute, therefore, comparisons are possible. Results are contrasted in Section 7.1.



Young Modulus	2.1 x $10^1$1 Pa
Poisson Modulus	0.2
Thickness	0.1 m
Maximum permissible stress	250×10^6 m

Fig. 9. Simply supported beam problem with a punctual load



Young Modulus	2.1 x $10^1$1 Pa
Poisson Modulus	0.2
Thickness	0.1 m
Maximum permissible stress	250×10^6 m

Fig. 10. Simply supported beam problem, discretized in 144 elements

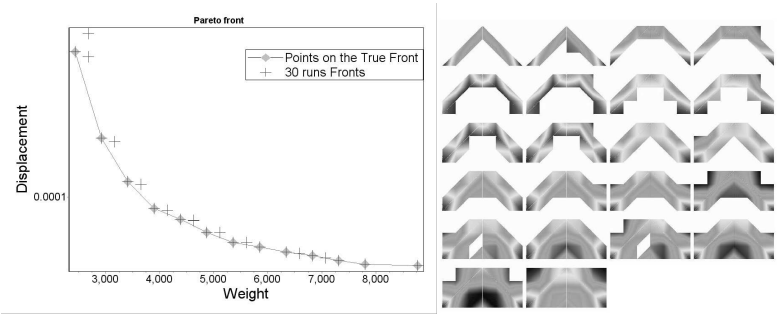
6.2 Experiment 2 (description)

Experiment 2 has the same conditions as experiment 1, as shown in Figure 10. But in this case the search space is discrete and consists of 144 elements. Since the true Pareto set is unknown, comparisons are made against a reference Pareto set obtained from 30 runs of our algorithm.

7 Results

7.1 Experiment 1. True front comparison

We compare results of 30 runs of our algorithm against the true Pareto front. There are 13 vectors in the true Pareto set. The average number of individuals found in the known Pareto set after 30 runs is 20.667, with an standard deviation of 1.3218 The average number of individuals from known Pareto set that belong to the true Pareto set is 12.6667 (out of 13), with a standard deviation of 0.479463. The average number of dominated individuals by the true Pareto set is 0.066667, with a standard deviation of 0.253708. In Figure 11, we can see the true Pareto front and the non-dominated structures found in 30 runs (the dominance is check after either independent run, so in the graph we can see some dominated individuals but they are not from the same run). Note that all 30 runs found all the structures in the true Pareto set.

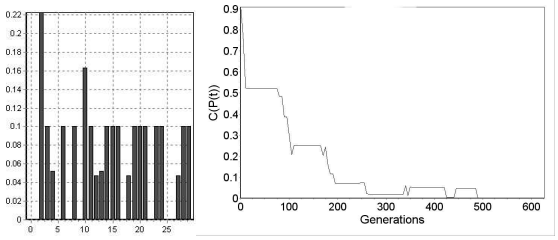


(a) **(b)**

Fig. 11. (a) Diamonds represent the true Pareto front; crosses are vectors found in 30 runs (30 fronts are plotted but they overlap with each other). (b) Structures found by a typical run of MASO.

7.2 Experiment 1. Convergence metric

We compute the convergence metric for the problem described in Section 6.1. Figure 12(a) shows the $C(P^{(t)})$ value for the last generation in 30 independent runs, the mean of 30 runs is 0.06783, with a standard deviation of 0.05586 (remember that smaller is better). On the right hand side, a convergence plot of a typical run is shown.



(a) **(b)**

Fig. 12. (a) $C(P^{(t)})$ in the last generation, (b) Convergence graph of a typical run of Experiment 1

7.3 Experiment 2. Reference front

Figure 13 shows the reference front and all the non-dominated individuals found in 30 runs. The crosses represent all the non-dominated individuals found in

30 runs. Although some of them are dominated by the reference Pareto front (shown with diamonds), they were not dominated in their independent run.

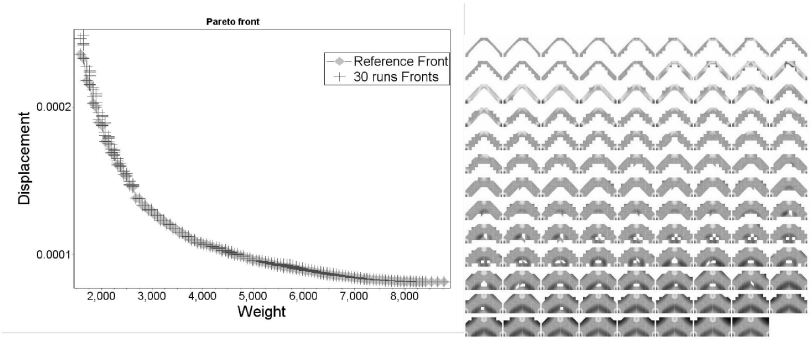


Fig. 13. (a) Diamonds are the reference front and the crosses represent the fronts from 30 runs, (b) Structures found by a typical run of MASO

7.4 Experiment 2. Convergence metric

Figure 14(a) shows the $C(P^{(t)})$ value for the last generation. As we can observe, the value is very close to 0 implying good convergence behavior. Figure 14(b) shows the convergence behavior to the front for a typical run. A decreasing distance is clear in the graph; at some points there are some peaks which mean that the algorithm found some individuals that dominate more than one non-dominated individual in the previous generation. The average convergence is 0.008312, with a standard deviation of 0.007347, for 30 independent runs.

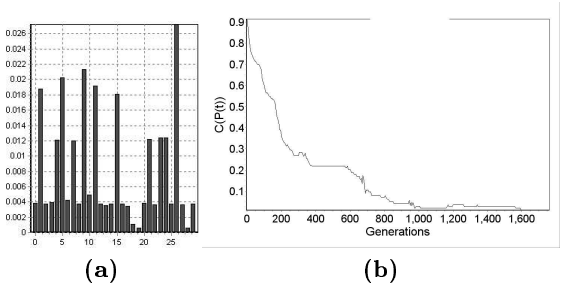


Fig. 14. (a) Graph of the convergence metric in 30 runs, (b) Convergence behavior

8 Conclusions

We proposed a new multiobjective optimization algorithm inspired by PBIL [7] and UMDA [6] algorithms. Several issues must be considered when using EDA algorithms, for example, the premature convergence of probability distributions. We proposed smoothing the probability distributions in order to improve exploration. The proposed method finds a set of structures that are optimal solution of the multiobjective problem, avoiding non-desired characteristics such as small holes and many objects. Even though we have suggested values for the different parameters used in the method, the algorithm is very robust to different parameter values.
It is worth to note that individuals on the Pareto front are equally spaced and spread all over. Also, note that the points that seem more spaced on the displacement axis, are equally spaced on the weight axis (because the relationship between weight and displacement is not linear).
The convergence metric makes evident the robustness of MASO. Even more, most vectors, 97.5% from the true Pareto set, were found by each run of a total of 30.

References

1. Chapman C.D., Saitou K., Jakiela M.J.. Genetic algorithms as an approach to configuration and topology design. Jou. Mech. Des. 116: 1005-11, (1994)
2. Deb K and Goell T. Multiobjetive Evolutionary Algorithms for Engineering Shape Optimization *KanGal report 200003 . Kanpur India*, (2000)
3. Kane, C. and Schoenauer M. Topological Optimum Design using Genetic Algorithms *Control and Cybernetics, Vol. 25 No. 5*, (1996)
4. Li, H., Zhang, Q., Tsang,E.P., and Ford, J.A. Hybrid Estimation of Distribution Algorithm for Multiobjective Knapsack Problem ,*In Proceedings of the 4th European Conference on Evolutionary Computation in Combinatorial Optimization, Coimbra, Portugal*, (2004)
5. Marroqun, J.L., Velasco,F.A., Rivera M. and Nakamura M. Gauss-Markov Measure Field Models for Low-Level Vision, *IEEE Trans. On PAMI, 23, 4: 337-348*, (2001)
6. Mühlenbein ,H. and PaaB, G. From reconbination of Genes to the estimation of distributions I. Binary parameters. *Parallel problem Solving form Nature (PPSN IV), 178-187*, (1996)
7. Shumeet Baluja. Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning *School of Computer Science Carnegie Mellon University, Pittsburgh, Pennsylvania 1523. CMU-CS-94-163* , (1996)
8. Pelikan M. Goldberg D.E and Cantu Paz. Linkage problem, distribution estimation and bayesian networks *IlliGal Report No. 98013 Urbana Il University*, (1998)
9. Pelikan M. Goldberg D.E and Cantu Paz. BOA: The Bayesian Optimization Algorithm *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, (1999)
10. L.E Malvern, *Introduction to the Mechanics of a Continuous Medium*, Pretence Hall Inc. Englewood Cliffs New jersey, (1969).

11. O.C. Zienkiewicz y R.L. Taylor: *El Método de los Elementos Finitos*; Cuarta Edición, Volumen 2, Mc. Graw Hill-CIMNE. (1995).
12. Kalyanmoy Deb and Sachin Jain. Running Perfomance Metrics for Evolutionary Multi-Objective Optimization *KanGal report 2002004 . Kanpur India*, (2000)