J Glob Optim (2007) 39:49–77
DOI 10.1007/s10898-006-9120-2

ORIGINAL PAPER

A new multiobjective simulated annealing algorithm

Ozan Tekinalp · Gizem Karsli

Received: 1 August 2005 / Accepted: 2 November 2006 / Published online: 6 December 2006

Abstract A new multiobjective simulated annealing algorithm for continuous optimization problems is presented. The algorithm has an adaptive cooling schedule and uses a population of fitness functions to accurately generate the Pareto front. Whenever an improvement with a fitness function is encountered, the trial point is accepted, and the temperature parameters associated with the improving fitness functions are cooled. Beside well known linear fitness functions, special elliptic and ellipsoidal fitness functions, suitable for the generation on non-convex fronts, are presented. The effectiveness of the algorithm is shown through five test problems. The parametric study presented shows that more fitness functions as well as more iteration gives more non-dominated points closer to the actual front. The study also compares the linear and elliptic fitness functions. The success of the algorithm is also demonstrated by comparing the quality metrics obtained to those obtained for a well-known evolutionary multiobjective algorithm.

Keywords Multiobjective optimization · Simulated annealing · Pareto front.

1 Introduction

Engineering design involves trading multiple objectives, concerning many different disciplines. After many trials, few feasible, but usually non-optimal designs emerge. To introduce optimality and automate the design process, multidisciplinary as well as multiobjective design optimization is needed. However, with multiple objectives, the solution to the problem is not unique. In the contrary, there are an infinite number of solutions. This process of optimizing many objective functions is called the

O. Tekinalp (✉) · G. Karsli
Aerospace Engineering Department, Middle East Technical University, 06531 Ankara, Turkey
e-mail: tekinalp@metu.edu.tr

Present Address:
G. Karsli
TUSAS Aerospace Industries, Ankara, Turkey

"multi-objective", "multicriteria" or "vector" optimization [3]. Edgeworth [3, 8] was the first to propose the notion of multiobjective optimization. The Italian economist Vilfredo Pareto generalized the multiobjective optimization in his book first published in 1896 [23]. For this reason, the hyper surface (or the curve, in bi-objective problems), containing the optimum solutions of multiple objectives is called the Pareto front.

Optimization algorithms are either deterministic, or stochastic. The deterministic ones requiring function derivatives, or gradient information, have been well developed. These algorithms have been adapted for multiobjective optimization. The usual approach is to use a scalarizing function to convert a vector of objectives to a scalar objective function. A comprehensive review of various continuous multiobjective methods that use gradient information may be found in [21]. However, gradient-based algorithms converge to local optimum. Since most engineering problems are modeled with nonlinear, multimodal and even discontinuous functions, gradient-based methods may fail to provide even a feasible solution. Among non-gradient optimization methods Evolutionary Algorithms (EAs) and Simulated Annealing (SA) have been most successful [11]. They have also been adapted for multiobjective optimization.

The population-based nature of EAs makes them quite suitable for multiobjective optimization. After applying evolutionary operations, such as cross over, mutation, and selection for sufficient number of generations, the final population shall contain mostly the points on the Pareto front. Many such algorithms have been developed for this purpose [7]. They have also been successfully applied to a number of real-world engineering problems as well [2, 6, 9, 13, 24]. However, the performance of the algorithm depends on the successful selection of various problem specific parameters. Thus, a particular set of parameters that are found to give excellent results for a particular problem may give very poor results for another problem (i.e., no free lunch).

The SA is an optimization method that simulates the physical annealing process of finding the low-energy states of a solid at a particular temperature. For a given temperature T, the probability of a system to be in state $\mathbf{r}$ may be found from the Boltzmann distribution $\exp\left(\frac{E(\mathbf{r})}{k_\mathrm{B}T}\right)$, where $E(\mathbf{r})$ is the energy of the configuration, and k_B is the Boltzmann constant [16]. To simulate the annealing process, Metropolis criterion may be used. For this purpose, the change in the energy of a system with the movement of an atom is calculated (ΔE). If the movement lowers the energy of the system, it is accepted ($\Delta E \leq 0$). Otherwise it is accepted with probability of $P(\Delta E) = \exp(-\Delta E/k_\mathrm{B}T)$. In the simulated annealing optimization method, the cost function replaces the energy of the system, and the optimization variables represent the atoms. This idea was first used by Kirkpatrick et al. to solve discrete combinatorial optimization problems [16]. The technique was later extended to the optimization of functions of continuous variables [1, 4, 10, 25, 27, 35].

In SA the success of the algorithm to find the global optimum by the fewest number of function evaluations is closely related to the method used in selecting the next candidate point. For this purpose, various methods are proposed. For example Vanderbilt and Lougie [35] uses random walk with a fixed maximum step size, where, for each variable, it is updated after a predetermined number of trials. The new set of step sizes is selected proportional to the inverse of the cost function's Hessian. Corona et al. on the other hand displaces one optimization variable at a time [4]. Siarry et al. also selects the next test point using random walk, but only displacing a randomly selected subset of optimization variables [27]. In these studies the step size is kept constant for a predetermined number of iterations [4, 27].

The cooling scheme used is also an important aspect of SA algorithms. The most common approach is to cool by multiplying the current temperature by a fixed factor after a predetermined number of trials [35] or records [4]. Hajek gives the necessary and sufficient conditions for a deterministic cooling schedule so that convergence to a global optimum is guaranteed [10]. Siarry et al. propose changing the temperature after a predetermined number of steps using two different factors for fast and slow cooling depending on the number of records found [27].

Hide-and-seek is a continuous SA algorithm that uses pure random walk, where both the search direction and step size are taken from uncorrelated uniform distributions, and an adaptive cooling schedule [1, 25]. The cooling is carried out whenever a new record is found. The temperature selected depends on the distance of the record from the estimated global optimum. Thus, the temperature is small if the current record is close to the estimated global optimum. Otherwise, it is large. The algorithm is proven to converge to the global optimum with probability one. It has been applied to trajectory optimization [20], and combined optimization of design and control variables [32, 33]. Changes to the basic algorithm that improves its convergence rate are proposed, and efficient ways to handle the equality constraints were examined [32].

SA has recently been adapted for the multiobjective combinatorial optimization problems [5, 30, 34]. The usual approach is to use a scalarizing function to convert a multiobjective problem to a single objective one. For this purpose, a weighted sum metric is considered [30, 34]. To achieve diversity, the program is run many times with different weight sets generated uniformly [34]. Czyżak and Jaszkiewicz proposed an algorithm, also for multiobjective combinatorial optimization problems, which uses a population of interacting solutions, called generating solutions [5]. Both scalar linear and weighted Chebyshev metrics were considered for the acceptance probability functions, while, the objective weights are changed according to whether there is an improvement in a particular objective function.

Suppaptnarm et al. proposed an SA based multiobjective algorithm, suitable for continuous optimization problems [29]. In this algorithm, each objective function is assigned a different temperature parameter. These parameters are cooled according to a rather complicated procedure that uses the standard deviation of the solutions accepted at a particular temperature value. Temperature parameters are updated periodically, as the number of iterations, or the number of records accumulated at a particular temperature exceeds the predetermined values. Another interesting aspect of the algorithm is the return to base strategy used to provide diversity to the solutions. The step size is also controlled and reduced in a random fashion, whenever a solution is accepted. However, to benefit from all these features, the user must supply a proper set of problem dependent parameters.

Suman proposed two multiobjective SA algorithms, called “Weight Based Multi Objective Simulated Annealing Algorithm,” and “Pareto Dominated Simulated Annealing Algorithm” [28]. The author also compared these algorithms to the three previously developed algorithms, for their effectiveness in solving number of benchmark problems on system reliability. The study has shown that some algorithms worked well with certain problems while performed poorly in other problems.

Kubotani and Yoshimura examined the effects of acceptance probability functions for multiobjective simulated annealing methods, as well as proposed a parameterized acceptance probability function [18]. They have shown that the solution quality is greatly affected by the shape of the acceptance probability function. The SA based multiobjective algorithms have also been applied to a number of multiobjective problems as well [19, 36].

The SA based multiobjective methods presented in the literature all have a number of problem dependent parameters to be decided. The applicability to most multiobjective problems has not been sufficiently demonstrated. Hide-and-Seek is a single objective SA algorithm that does not have parameters related to cooling schedule or search method, and has been shown to rapidly converge to the global optimum for multi-modal and nonlinear problems [1, 20, 25, 32, 33]. For this reason, it is decided to develop a multiobjective algorithm based on Hide-and-Seek.

A new SA based multiobjective algorithm, to be used for continuous optimization problems is presented in this manuscript. It is shown that the algorithm is capable of obtaining many points very close to the actual Pareto front.

In the next section, the multiobjective optimization problem is described together with some formulation approaches especially used in SA based algorithms. Then, the new Multiple Cooling MultiObjective Simulated Annealing (MC-MOSA) algorithm is given, followed by the presentation of special elliptic and ellipsoidal fitness functions suitable for non-convex fronts. In the results and discussion section, five problems are solved to demonstrate the effectiveness of the algorithm. A parametric study is also presented to examine the effect of two critical parameters: the number of fitness functions used and number iterations carried out in obtaining the Pareto front. The importance of fitness function type, linear or elliptic, is also examined in the parametric study section. A comparative study, to examine the success of MC-MOSA algorithm against the popular genetic-based multiobjective algorithm NSGA-II, is also presented. Finally conclusions are given.

2 Formulation of the Problem

A multiobjective optimization problem may be stated as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{Minimize}} \quad & f_i(\mathbf{x}), && i = 1, \ldots, I \\ \text{subject to:} \quad & g_a(\mathbf{x}) \geq 0, && a = 1, \ldots, A, \\ & h_b(\mathbf{x}) = 0, && b = 1, \ldots, B, \\ & x_j^L \leq x_j \leq x_j^u, && j = 1, \ldots, J. \end{aligned} \tag{1}$$

In most problems, some objectives are maximized while others are minimized. The maximization requests may be handled by minimizing the negatives of the objective functions to be maximized [7]. The above vector optimization problem may be converted into a scalar optimization problem using a scalarizing function. Many scalarizing function have been developed for this purpose. A general scalarizing function may be expressed as [7]:

$$F = \left(\sum_{i=1}^{I} v_i \left| f_i(\mathbf{x}) - f_i^0 \right|^{\beta} \right)^{1/\beta}. \tag{2}$$

In the above expression, f_i^0 are the ideal solutions, v_i is the weight associated with the ith objective function ($v_i \geq 0$), while β is a parameter ($1 \leq \beta \leq \infty$). When $\beta = 1$, we have the weighed sum approach, while $\beta = 2$ gives the usual weighted Euclidian norm.

For large β, the formulation reduces to minimizing the largest deviation of $\left|f_i(\mathbf{x}) - f_i^0\right|$, called the weighted Chebyshev metric [7]. Due to the difficulty of assigning weights to objectives of different magnitude, it is advisable to normalize the objective functions:

$$\tilde{f}_i(\mathbf{x}) = \frac{f_i(\mathbf{x}) - f_i^{\min}}{f_i^{\max} - f_i^{\min}}, \tag{3}$$

where $f_i^{\max}$ and $f_i^{\min}$ refer to the maximum and minimum values of the ith objective in the feasible space. Then Eq. 2 may be written as:

$$\tilde{F} = \left(\sum_{i=1}^{I} w_i \left|\tilde{f}_i(\mathbf{x}) - \tilde{f}_i^0\right|^{\beta}\right)^{1/\beta}. \tag{4}$$

In the above expression, w_i, are the related weight coefficients of the normalized objectives.

Like most non-gradient algorithms, the SA algorithms are also developed for unconstrained optimization. The common approach is to augment the constraints to the cost function using penalty coefficients. In this manuscript the following is employed:

$$\Omega = -\sum_{a=1}^{A} \eta_a \min(0, g_a(\mathbf{x})) + \sum_{b=1}^{B} \xi_b \left|h_b(\mathbf{x})\right|, \tag{5}$$

where, Ω is the total cost due to constraints, and, η_a and ξ_b are the related penalty coefficients associated with each constraint. The scalarizing function together with the cost due to constraints may be written as,

$$\tilde{F} = \left(\sum_{i=1}^{I} w_i \left|\tilde{f}_i(\mathbf{x}) - f_i^0\right|^{\beta}\right)^{1/\beta} + \Omega. \tag{6}$$

Then, the constrained multiobjective optimization problem is reduced to a multiobjective problem with bounds on optimization variables:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{Minimize}}\ \tilde{F}(\mathbf{x}), \quad i = 1, \ldots, I, \\ &\text{subject to: } x_j^L \le x_j \le x_j^U, \quad j = 1, \ldots, J. \end{aligned} \tag{7}$$

2.1 Multiobjective simulated annealing algorithms

The acceptance probability used in SA algorithms shall be re-considered to introduce multiobjectiveness. When there are many objectives, three cases may be encountered. First the test point **y**, improves all the objectives as compared with the current solution. In this case the test point **y** shall be accepted. Second, some objective function values improve, while others deteriorate. Since, the test point **y**, and the current solution **x**, does not dominate each other the test point **y** may again be accepted based on the concept of non-dominance. Finally, all function values deteriorate. However, deteriorating points shall also be accepted with probability. For this reason, the usual approach in SA is to employ a scalarizing function in evaluating acceptance probabilities. The most common acceptance probabilities employed are presented below [18].

2.1.1 Scalar linear rule

This uses the weighted sum scalarizing function. Using the normalized objective functions (Eq. 3), the acceptance probability may be written as:

$$Pr = \min\left\{1, \exp\left(\sum_{i=1}^{I} w_i\, \Delta\tilde{f}_i \Big/ T\right)\right\}, \tag{8}$$

where, T is the temperature, $\Delta\tilde{f}_i = \tilde{f}_i(\mathbf{x}) - \tilde{f}_i(\mathbf{y}), \tilde{f}_i(\mathbf{x})$ are the set of current objective values, while, $\tilde{f}_i(\mathbf{y})$ are the objective functions evaluated at test point $\mathbf{y}$, and $\sum_{i=1}^{I} w_i = 1$.

2.1.2 Chebyshev rule

In this worst-case rule, if all objective function values at the test point improve the test point is accepted with probability one. Otherwise, the smallest acceptance probability value is used. Thus,

$$Pr = \min\left\{1, \min_i\left\{\exp\left(\frac{w_i \Delta\tilde{f}_i}{T}\right)\right\}\right\}. \tag{9}$$

2.1.3 Weak rule

In this case, if any of the objective functions improve, then the test point is accepted with probability one. Otherwise the highest probability acceptance value is used. Thus,

$$Pr = \min\left\{1, \max_i\left\{\exp\left(\frac{w_i \Delta\tilde{f}_i}{T}\right)\right\}\right\}. \tag{10}$$

In general, different probability acceptance function may be written using different scalarizing functions (Eq. 6):

$$Pr = \min\left\{1, \left\{\exp\left(\frac{\Delta\tilde{F}}{T}\right)\right\}\right\}, \quad \Delta\tilde{F} = \tilde{F}(\mathbf{x}) - \tilde{F}(\mathbf{y}). \tag{11}$$

3 Multiple cooling multiobjective simulated annealing algorithm

The main deficiency of the above approach is its single objective nature. For this reason, the true minimum of a scalarizing function, or more appropriately a *fitness function* (FF) (Eqs. 2, 4, and 6), is a particular point on the Pareto front. For example, if a weighted sum of the objectives is minimized the true minimum, F^*, is a single point on a convex feasible region boundary, where the line defined by the weighted sum is tangent as shown in Fig. 1. To properly define the Pareto surface, many fitness functions may be used. In this case each function has a different minimum point, F_j^*, on the Pareto front (Fig. 2). For this reason, some previous algorithms conducted a separate run with each fitness function. Consequently, the total number of iterations required was quite high. What is needed is not to minimize a single fitness function at a time, but to minimize a population of fitness functions together. Thus, a

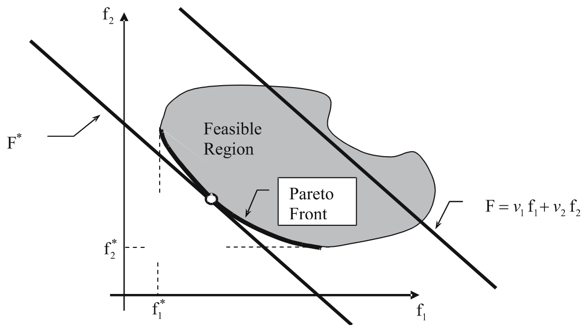


Fig. 1 Pareto front, and the minimum value of the weighted sum, F^*, on the Pareto front

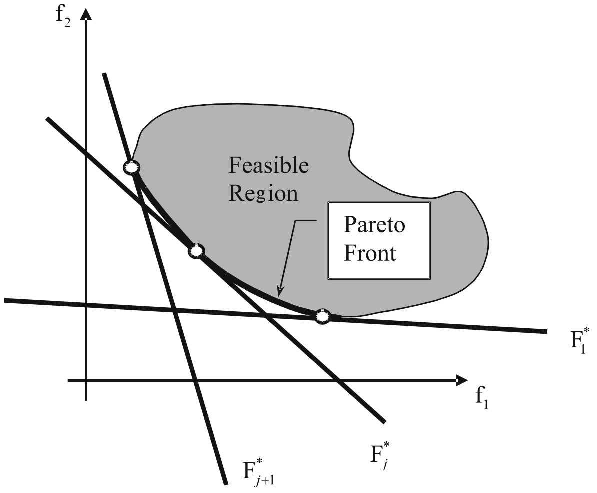


Fig. 2 The family of weighted sum fitness functions used in MC-MOSA with their minimum values on the Pareto front

population-based approach is proposed in this manuscript. For example, a population containing weighted sum type (linear) FFs with added cost due to constraints, may be written as:

$$\begin{gathered}\tilde{F}_m = \sum_i^I w_{mi}\tilde{f}_i(\mathbf{x}) + \Omega_m, \quad m = 1, \ldots, M, \\ \sum_i^I w_{mi} = 1.\end{gathered} \tag{12}$$

Another novelty of the proposed algorithm is the assignment of a separate temperature parameter, T_j, to each fitness function, $\tilde{F}_j$. Each temperature, T_j, is cooled individually, if the value of the related fitness, $\tilde{F}_j$, has improved (or reduced in the minimization problem). The test point is accepted based on the highest value of the probability of acceptances. Thus, a test point that improves any fitness function is accepted, since, it should be close to the Pareto front. Those test points that cause deteriorations are accepted based on the lowest deterioration value (or highest probability acceptance value) as well. In this way, instead of moving towards a particular point on the front, by accepting points close to the front with high probability, the procedure moves towards the whole Pareto front. The algorithm, just like the single objective simulated annealing algorithm Hide-and-Seek [1, 25], uses pure random walk to generate the next test point, and an adaptive cooling schedule.

3.1 The MC-MOSA algorithm

Step 0 Initialize random number generators; generate the initial test point $\mathbf{x}^0 \in S$ and set elements of the temperature vector, $\mathbf{T}^0$ of dimension M (i.e., number of FFs to be used), to high values. Initialize the loop counter ($K = 0$). Also initialize the best and next best records of the FFs ($\tilde{\mathbf{F}}^{\text{best}} = \tilde{\mathbf{F}}^{\text{nextbest}} = \mathbf{0}$).

Step 1 Choose a search direction $\boldsymbol{\theta}^K$on the surface of a unit sphere with uniform distribution. Also choose a step size λ^K from uniform distribution such that $\Lambda^K = (\lambda^K \in R;\ \mathbf{x}^K + \lambda^K\, \boldsymbol{\theta}^K \in S)$ set $\mathbf{y}^K = \mathbf{x}^K + \lambda^K\, \boldsymbol{\theta}^K$.

Step 2 Generate $V^K (0 \le V^K \le 1)$ from uniform distribution.

Step 3 Evaluate the probability acceptance function

$$\begin{gathered} Pr = \min\left\{1, \max_m \left[\exp\left(\Delta\tilde{F}_m^K / T_m^K\right)\right]\right\}, \\ \Delta\tilde{F}_m^K = \tilde{F}_m(\mathbf{x}^K) - \tilde{F}_m(\mathbf{y}^K), \quad m = 1, \ldots, M, \end{gathered} \tag{13}$$

where $\tilde{F}_m$ is a set of linear (Eq. 12) or higher order fitness functions (Eq. 6).

Step 4 Accept the trial point, $\mathbf{y}^K$, with probability Pr

$$\mathbf{x}^{K+1} = \begin{Bmatrix} \mathbf{y}^K & \text{if } V^K \in (0,\ Pr) \\ \mathbf{x}^K & \text{otherwise} \end{Bmatrix}. \tag{14}$$

Step 5 If $Pr = 1$ (i.e., if there are any improving fitness functions, $\tilde{F}_m(\mathbf{y}^K)$, ($m = 1, \ldots, M$)):

Step 5.1 Archive the test point ($\mathbf{x}^{K+1} = \mathbf{y}^K$), as well as values of the objectives, ($\mathrm{f}_i(\mathbf{y}^K)$), to be further processed to obtain the Pareto front.

Step 5.2 For those FFs that have improved at the test point:

(1) Update the best, and next best records related to the improving FF (i.e., $\tilde{F}_m^{\text{nextbest}} = \tilde{F}_m^{\text{best}}$, and $\tilde{F}_m^{\text{best}} = \tilde{F}_m(\mathbf{y}^K)$).

(2) Update the related temperature using,

$$T_m = 2[\tilde{F}_m(\mathbf{x}^{K+1}) - \tilde{F}_m^*]/\chi^2_{1-p}(d). \tag{15}$$

In this formula, $\tilde{F}_m^*$, is the global minimum of the mth fitness function, and $\chi^2_{1-p}(d)$ is the $100(1-p)$ percentile point of the chi-square distribution with d degrees of freedom. Since this global minimum is not known in advance, its estimate, $\tilde{F}_m^e$, is used instead [1, 25]: $\tilde{F}_m^e = \tilde{F}_m^{\text{best}} + \frac{\tilde{F}_m^{\text{best}} - \tilde{F}_m^{\text{nextbest}}}{(1-p)^{-d/2} - 1}$. As shown in [32] using the estimator with the expected upper and lower bounds results in a better estimator and improves the convergence rate of Hide-and-Seek simulated algorithm. Consequently, the estimator may also be used with upper and lower bounds in this algorithm as well:

$$\tilde{F}_m^* = \begin{cases} \tilde{F}_m^{\text{lower}}, & \text{if, } \tilde{F}_m^e < \tilde{F}_m^{\text{lower}}, \\ \tilde{F}_m^{\text{upper}}, & \text{if, } \tilde{F}_m^e > \tilde{F}_m^{\text{upper}}, \\ \tilde{F}_m^e, & \text{otherwise.} \end{cases} \tag{16}$$

Step 6 Increment the loop counter and go to Step 1 and loop until permitted number of function evaluations is completed. One may also employ additional conditions for terminating the loop, such as terminating if all the temperatures have dropped below a pre-determined value, or no trial point has been accepted (i.e., a very low-*Pr* value) for a predetermined number of iterations.

Step 7 To get a set of solutions that approximates the actual Pareto front, carry out a non-dominated selection from the results archived in Step 5.1.

The linear fitness functions constructed using weighted sums, are especially suitable for convex Pareto fronts. For non-convex problems, scalarizing function based on weighted Chebyshev metric is theoretically shown to be the only function to capture the whole Pareto front [21]. However, in the following section, special elliptic and ellipsoidal FFs are introduced, which is expected to have a good performance in capturing non-convex fronts.

3.2 Elliptic fitness function

Consider an ellipse centered at $C\left(\tilde{f}_1^C,\tilde{f}_2^C\right)$, while semi major and semi minor axis are aligned with the coordinate directions. If the ellipse passes through point $P\left(\tilde{f}_1^P,\tilde{f}_2^P\right)$, its semi major axis may be written as:

$$a = \sqrt{(\tilde{f}_1^P - \tilde{f}_1^C)^2 + \kappa(\tilde{f}_2^P - \tilde{f}_2^C)^2}, \tag{17}$$

where, $\kappa = 1/(1 - e^2)$, and e is the eccentricity of the ellipse. If the semi major axis is aligned with a line connecting ellipse center C to point $G(\tilde{f}_1^G,\tilde{f}_2^G)$, then the semi major axis of this ellipse may be written as:

$$a = \sqrt{\mathbf{r}^{\mathbf{T}}\mathbf{Q}^{\mathbf{T}}\mathbf{\Lambda}\mathbf{Q}\mathbf{r}}, \tag{18}$$

where,

$$\mathbf{r} = \left\{\tilde{f}_1^P - \tilde{f}_1^C, \tilde{f}_2^P - \tilde{f}_2^C\right\}^T, \quad \mathbf{Q} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} 1 & 0 \\ 0 & \kappa \end{bmatrix}, \tag{19}$$

where, φ is the angle between the horizontal axis and the line $\overline{CG}$ (Fig. 3).

A family of ellipses may be constructed by uniformly spreading their centers along a quarter circle Fig. 3. Then, the objective is to minimize the semi major axes of the ellipses (i.e., $F_j = a_j$). The minimum of each semi major axis becomes the point closest to a particular center.

3.3 Ellipsoidal fitness function

The above ides may easily be extended for a three-dimensional problem. The semi major axis of an ellipsoid that has a circular cross section in the f_2f_3-plane, centered at origin and passes through point $P\left(\tilde{f}_1^P,\tilde{f}_2^P,\tilde{f}_3^P\right)$, may be written as:

$$a_{3d} = \sqrt{(\tilde{f}_1^P)^2 + \kappa(\tilde{f}_2^P)^2 + \kappa(\tilde{f}_3^P)^2}. \tag{20}$$

Similarly, an ellipsoid centered at $C(\tilde{f}_1^C,\tilde{f}_2^C,\tilde{f}_3^C)$ with semi major axis directed toward point $G(\tilde{f}_1^G,\tilde{f}_2^G,\tilde{f}_3^G)$

$$a_{3d} = \sqrt{\mathbf{r}_{3d}^{\mathbf{T}}\mathbf{Q}_{3d}^{\mathbf{T}}\mathbf{\Lambda}_{3d}\mathbf{Q}_{3d}\mathbf{r}_{3d}}, \tag{21}$$

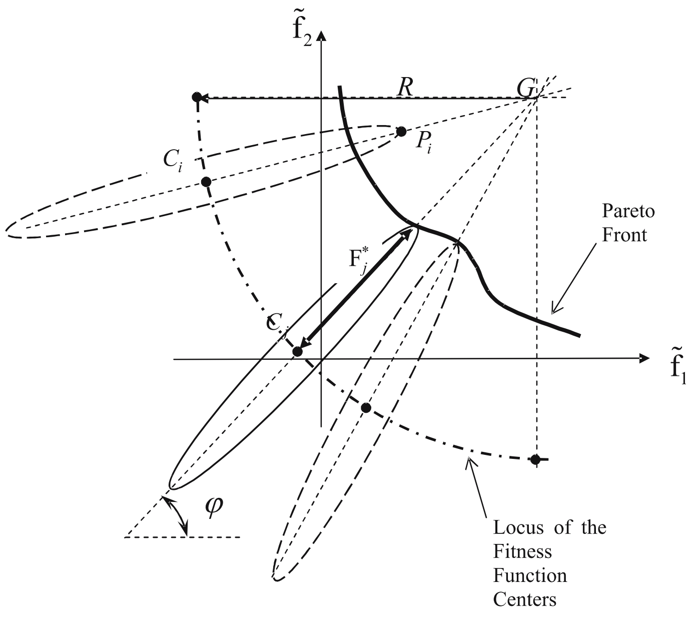


Fig. 3 Elliptic fitness functions

where,

$$\mathbf{r}_{3d} = \left\{\tilde{f}_1^P - \tilde{f}_1^C, \tilde{f}_2^P - \tilde{f}_2^C, \tilde{f}_3^P - \tilde{f}_3^C\right\}^T, \quad \mathbf{\Lambda}_{3d} = \text{diagonal}(1, \kappa, \kappa) \tag{22}$$

To align the $\tilde{f}_1$-axis with the line connecting point C to the center of the sphere, G, a sequence of two rotations may be used: First, rotate by an angle 'φ' along the $\tilde{f}_3$-axis, then rotate by an angle 'θ' around the negative $\tilde{f}_2$−axis direction. The transformation matrix then becomes:

$$\mathbf{Q}_{3d} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{23}$$

To obtain a family of ellipsoidal fitness functions a set of uniformly distributed points on the surface of an eighth of a sphere are generated. Although positioning of a uniformly distributed set of points on a sphere is a rather difficult problem, for the purposes of the optimization algorithm proposed in this manuscript, a perfect uniformity is not necessary. Consequently, an almost uniform set of points is automatically generated using an algorithm developed for this purpose.

4 Results and discussion

The first two problems presented below are taken from [7], while the following two are examples of a non-convex optimization and a three objective optimization. The fifth problem is an aerospace application, where the goal is to find the best trajectories of a launch vehicle that inserts a particular payload into an elliptic orbit maximizing both the apogee and the perigee.

4.1 Test Problem 1 (TP1): structural optimization

The first problem is on the design of a two-bar truss structure as shown in Fig. 4. The objectives are to minimize the material volume, as well as the maximum stress on the truss members. There are also bounds on the geometric dimensions and maximum stress allowed [7], while the bars are assumed to be rigid.

$$\text{objectives:}\begin{cases}\text{minimize } f_1 = A_1\sqrt{16+y^2} + A_2\sqrt{1+y^2}\\ \text{minimize } f_2 = \max(\sigma_{AC}, \sigma_{BC})\end{cases}$$

$$\text{constraints:}\begin{cases}\max(\sigma_{AC}, \sigma_{BC}) \le 10^5\,\text{kPa},\\ 1 \le y \le 3\,\text{m},\\ 0 \le A_1, A_2 \le 0.01\,\text{m}^2.\end{cases} \tag{24}$$

TP1 is first solved using a single FF with equal weights on normalized objectives. This case is abbreviated as MOSA in the following discussions.

$$\text{minimize}(w_1\tilde{f}_1 + w_2\tilde{f}_2 + \Omega), \quad w_1 = w_2 = 0.5. \tag{25}$$

The results plotted Fig. 5, are obtained after ten thousand (10k) iterations show that the points found are at the middle section of the front as expected. Since a single weighted sum converges only to a particular point, it should be possible to find different parts of the front with different weight sets. To demonstrate this behavior, the optimization is repeated seven times, using seven equally spaced weight values (i.e., $M = 7$), and executing for 10k iterations in each run.

$$\begin{gathered}\tilde{F}_m = w_{m1}\tilde{f}_1 + w_{m2}\tilde{f}_2 + \Omega\\ w_{m1} = (m-1)/M - 1\,, \qquad w_{m2} = 1 - w_{m1}, \quad m = 1, \ldots, M.\end{gathered} \tag{26}$$

The results are plotted in Fig. 6, where the points found with each set of weights are depicted with a different shape. It may be observed from this figure that, in this case, there are many more points that spreads well through out front.

In Fig. 7, MC-MOSA results obtained using five linear FFs ($M = 5$, in Eq. 26) and after 10k iterations are presented. It may be seen from the figure that not only there

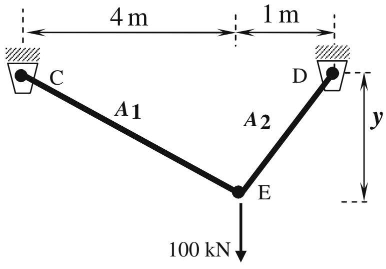


Fig. 4 Properties of the two bar truss of TP1

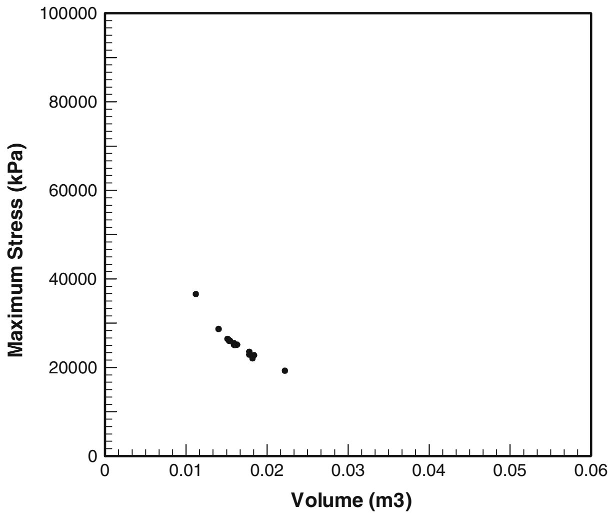


Fig. 5 MOSA results of TP1 after 10k iterations

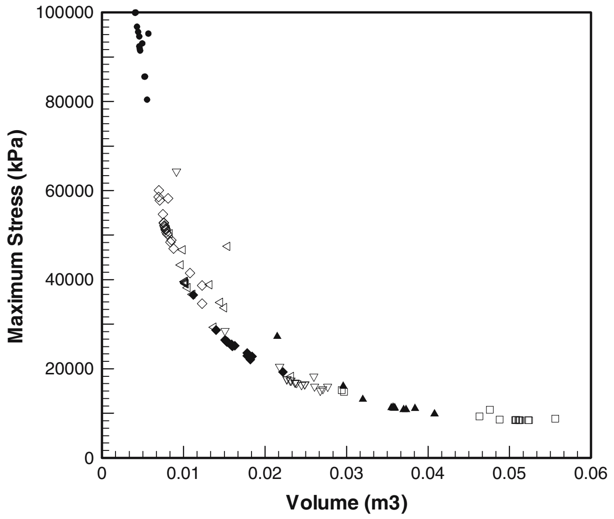


Fig. 6 MOSA results of TP1, after 7, 10k runs with different weights sets

are many points on the front, but the points found spreads well to the whole front as well. The optimization results using nine linear FFs ($M = 9$, in Eq. 26) are shown in Fig. 8. These results are again obtained after ten thousand iterations. It may be seen from the figure that, with 9-FFs, many more points are obtained on the front, than that obtained by 5-FFs. The points found spreads more uniformly on the front. Better fronts may be obtained by increasing the number of fitness functions used.

A comparative plot to show the effect of the iteration number in obtaining the Pareto front (Fig. 9) is obtained using 9-FFs. It may also be observed from the figure

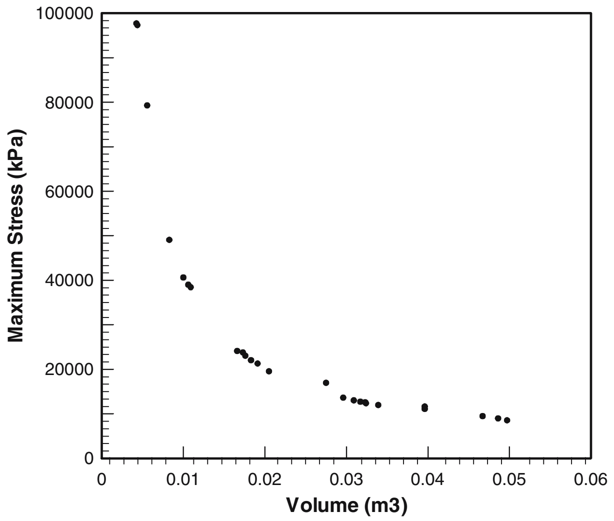


Fig. 7 TP1 using five linear FFs and after 10k iterations

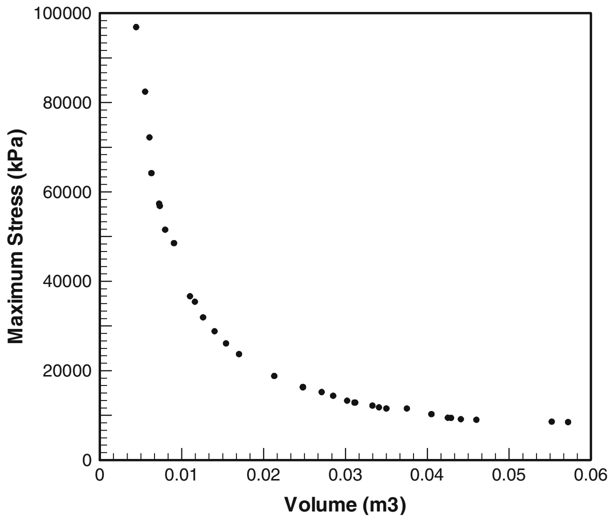


Fig. 8 TP1 using 9-FFs and after 10k iterations

that as the number of iterations increase, not only many more points are obtained, they are also closer to the actual front. These issues are examined in detail in the parametric study section below.

A close examination of the problem shows that the two ends of the front are at $\mathbf{f} = (0.004\,\mathrm{m}^3, 100\,\mathrm{MPa})^T$, and $\mathbf{f} = (0.051\,\mathrm{m}^3, 8.5\,\mathrm{MPa})^T$. The extremes of the front, shown in Fig. 8, are at $\mathbf{f} = (0.0042\,\mathrm{m}^3, 96.88\,\mathrm{MPa})^T$, and $\mathbf{f} = (0.057\,\mathrm{m}^3, 8.50\,\mathrm{MPa})^T$, respectively.

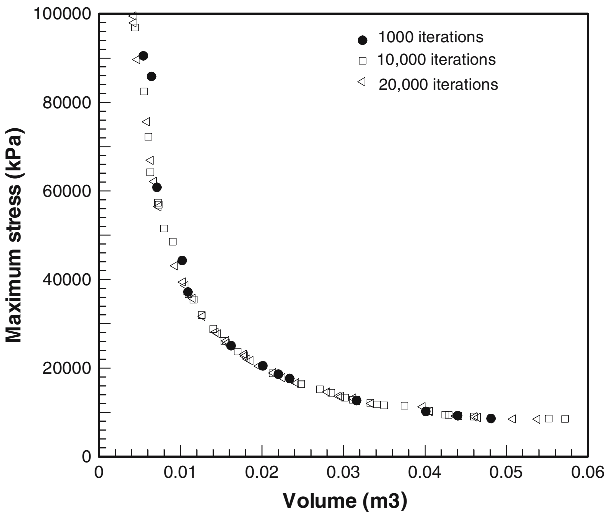


Fig. 9 TP1 with nine linear FFs, with different number of iterations

4.2 Test Problem #2 (TP2) 7

$$\text{objectives:} \begin{cases} \text{minimize } f_1 = 2 + (x_1 - 2)^2 + (x_2 - 1)^2, \\ \text{minimize } f_2 = 9x_1 - (x_2 - 1)^2 \end{cases}$$

$$\text{constraints:} \begin{cases} g_1 = x_1^2 + x_2^2 \leq 225, \\ g_2 = x_1 - 3x_2 + 10 \leq 0, \\ -20 \leq x_1 \leq 20, \\ -20 \leq x_2 \leq 20. \end{cases} \tag{27}$$

The permitted number of iterations was selected to be 10k, since in [7], the problem was solved using a population size of 100 and for 100 generations. The Pareto points plotted in Fig. 10 are obtained using a single linear fitness function with equal weights (Eq. 25). The two inequality constraints are also normalized and added to the cost with a penalty coefficient of 10 (Eq. 6).

The problem is also solved using the MC-MOSA algorithm. For this purpose, 9-FFs, constructed using nine equally spaced weight sets are employed ($M = 9$ in Eq. 26). The multiobjective optimization is again carried out for 10k function evaluations. The results are plotted in Fig. 11. It may be seen from these results that, there are many points on the front. Both lower right and upper left parts of the front stretches to the boundaries of the feasible region. A close examination of the problem reveals that the upper left corner is around $\mathbf{f} = (10.1, 2.6)^T$, while lower right corner is around $\mathbf{f} = (223.7, -217.7)^T$. The two ends of the front given in Fig. 11 are at $\mathbf{f} = (10.78, 1.52)^T$, and $\mathbf{f} = (223.95, -217.23)^T$, which are very close to the extreme values of the actual front. The front between these two extremes is almost a straight line (bounded by $x_1 = -2.5$, i.e., $f_2 + f_1 - 0.25 = 0$).

The linear fitness functions are especially suitable for convex fronts. For this reason, few points are obtained along the straight-line portion of the front, while the upper left part is well populated (see [7] for the plot the feasible region). This problem is also solved very successfully using elliptic FFs, while the results are presented in the parametric study section below.

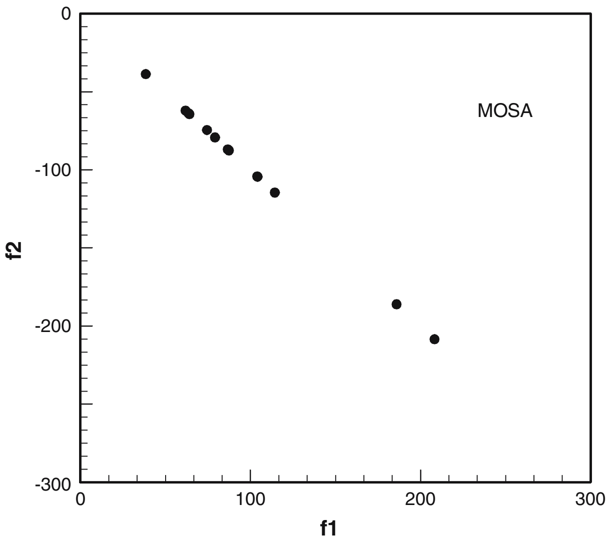


Fig. 10 MOSA algorithm results for TP2 after 10k iterations

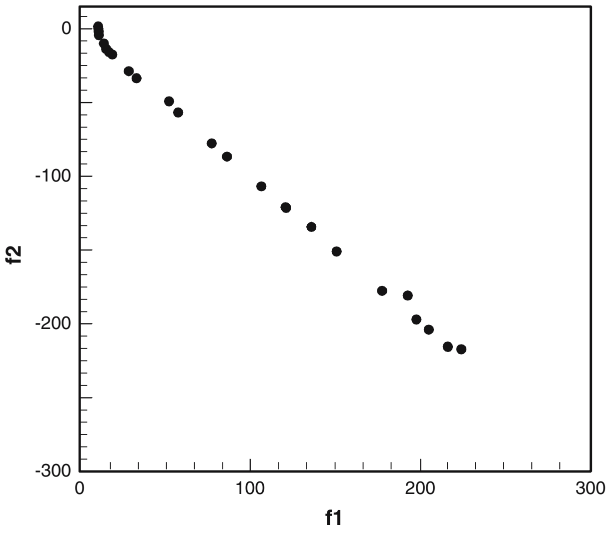


Fig. 11 TP2 results with MC-MOSA using nine linear FFs and after 10k iterations

4.3 Test Problem 3 (TP3): non-convex optimization

This multiobjective optimization problem is defined as follows [22]:

$$\text{objectives:} \begin{cases} \text{mimimize } f_1 = x, \\ \text{mimimize } f_2 = y, \end{cases}$$
$$\text{constraints:} \begin{cases} 5e^{-x} + 2e^{-0.5(x-3)^2} - y \leq 0, \\ 0 \leq x \leq 5, \\ 0 \leq y \leq 5. \end{cases} \tag{28}$$

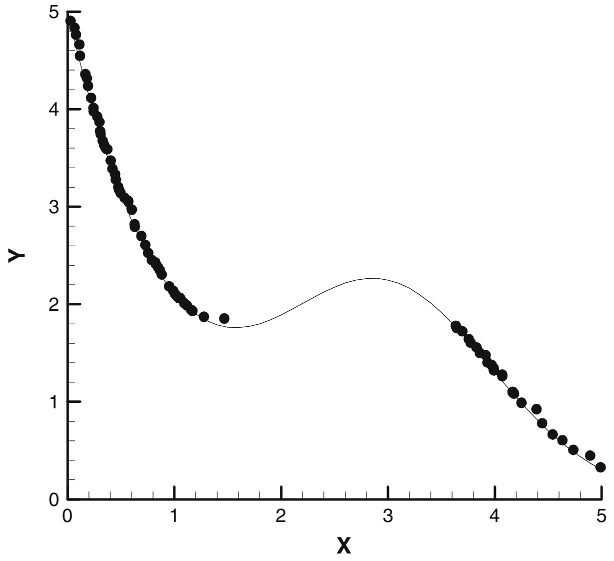


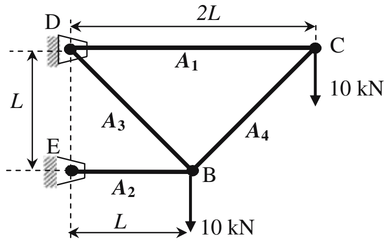
Fig. 12 TP#3: Non-convex optimization. 20k iterations, 250 elliptic FFs, $e = 0.999$, $R = 2$. The front contains 79 non-dominated points

The lower left-hand side boundary of the feasible region is the curve specified: $5e^{-x} + 2e^{-0.5(x-3)^2} - y = 0$. In Fig. 12, this boundary as well as the non-dominated points obtained with MCMOSA using 250 elliptic FFs and after 20k function evaluations are given. Centers of the elliptic fitness functions are distributed uniformly on a quarter circle, centered at (1,1) with a radius of 2. As shown in Fig. 12 the boundary of the feasible region is quite non-convex. Note that there are no points along the concave part of the boundary. Since those points are dominated by the neighboring solutions, they are removed from the set after the non-dominated sorting process.

4.4 Test problem 4 (TP4): a three objective optimization example

In this section, we examine a three objective optimization problem taken again from [22]. The problem is a four bar truss problem as shown in Fig. 13. The objectives are to minimize the stress in bars 1 and 4 while minimizing the total material volume at the same time. About 10 kN load is applied to nodes B and C. The cross sectional areas of

Fig. 13 The geometry of the four bar truss problem used in the three objective optimization example [22]



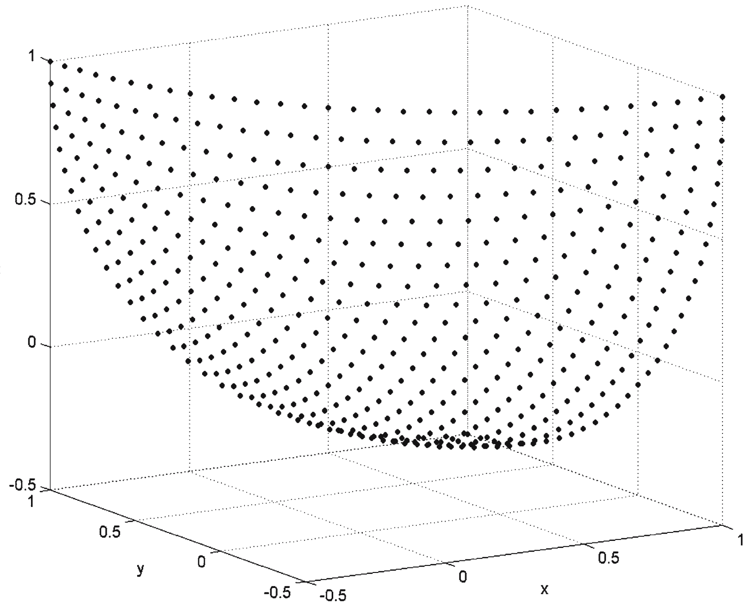


Fig. 14 Locations of the centers of 496 ellipsoidal FFs used in TP5

the bars are not to exceed 5 cm^2, and the stresses in the bars are limited to 10 kN/cm^2.

$$\text{objectives:} \begin{cases} \text{mimimize } f_1 = \sigma_1, \\ \text{mimimize } f_2 = \sigma_2, \\ \text{mimimize } f_3 = \text{Volume}, \end{cases} \tag{29}$$
$$\text{constraints:} \begin{cases} \max(A_1, A_2, A_3, A_4) \leq 5\,\text{cm}^2, \\ \max(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \leq 10\,\text{kN/cm}^2. \end{cases}$$

The optimization variables are the cross sectional areas of the bars (A_i). Since the problem is statically determinate, the forces in the bars may be calculated. To minimize the material volume, A_3 and A_4, may be chosen to load them to their maximum stress capacity (i.e., $\sigma_3 = \sigma_4 = 10\,\text{kN/cm}^2$). Then there are two optimization variables:A_1 and A_2.

For this problem, ellipsoid centers' are distributed on an eighth of a sphere surface, centered at (1,1,1) and have a radius of 1.5. The locations of those 496 automatically generated ellipsoid centers are plotted in Fig. 14. The semi major axes of the ellipsoids were directed toward (1,1,1) point. The points obtained after 20k iterations are shown in Fig. 15. In this figure, those 1,258 non-dominated points approximating the Pareto surface is quite visible. Although a 5 k run resulted in 735 points on the front, the edges of the surface was not as clear as the surface depicted in Fig. 15, and is not presented here.

4.5 Launch vehicle trajectory optimization example

This last example is on the trajectory optimization of a highly nonlinear launch vehicle model with 28 optimization variables. A multiobjective trajectory optimization

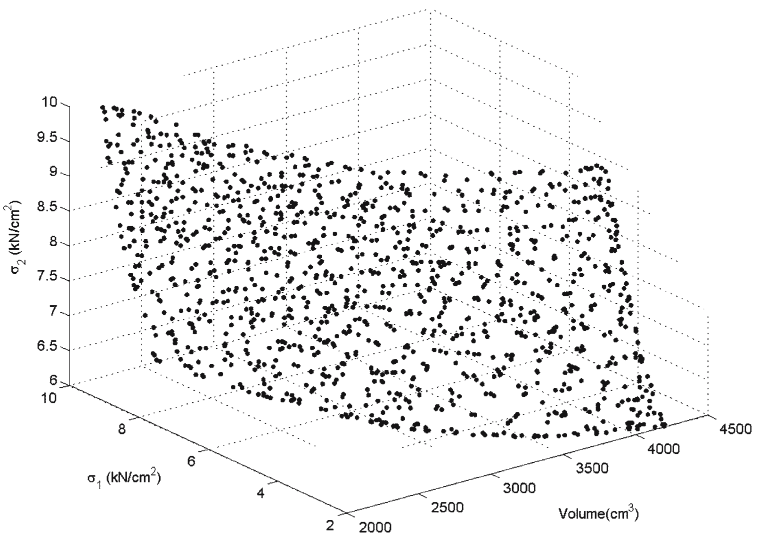
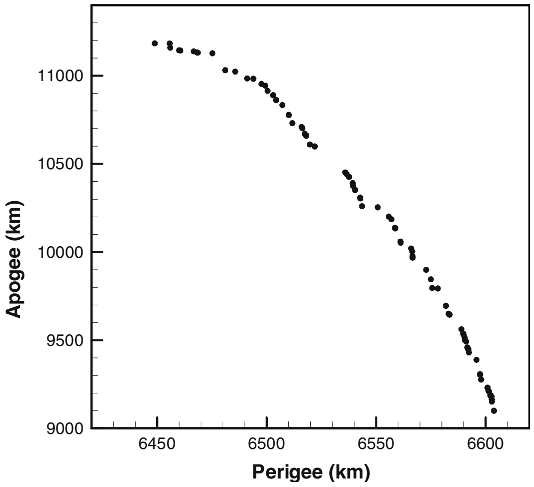


Fig. 15 TP4, optimization with three objectives. Figure obtained after 20k iterations with 496 ellipsoidal FFs. The figure contains 1258 non-dominated points

Fig. 16 TP5: Launch vehicle results with nine linear FFs and 100k iterations



problem may be stated as follows [32] Fig. 16:

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f_i(\mathbf{x}), \quad i = 1, \ldots, I, \tag{30}$$

$$\begin{gathered} \dot{\mathbf{x}} = \boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}, t), \\ \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \geq \mathbf{0}, \qquad \mathbf{h}(\mathbf{x}, \mathbf{u}, t) = \mathbf{0}, \end{gathered} \tag{31}$$

$$\begin{gathered} \mathbf{x}(t_0) = \mathbf{x}_0, \\ \boldsymbol{\Psi}(\mathbf{x}_f, \mathbf{u}_f, t_f) = 0 \qquad \boldsymbol{\varphi}(\mathbf{x}_f, \mathbf{u}_f, t_f) \leq 0. \end{gathered} \tag{32}$$

The problem is to find the control history, $\mathbf{u}(t)$, such that set of performance indices, f_i, are minimized (Eq. 30) subject to differential equality constraints, as well as algebraic equality and inequality constraints on state and control variables (Eq.31), with specified initial conditions and constraints on terminal conditions (Eq. 32). The trajectory optimization problems are optimal control problems traditionally solved using calculus of variations. Calculus of variations yields a two-point boundary value problem, which usually requires numerical techniques to solve. Another method is to convert the optimal control problem into a parameter optimization problem [14, 32]. For this purpose the value of the inputs ($\mathbf{u}_n$) discrete points in time ($t_n,\ n = 1, \ldots, N$), called nodes, are sought for. Two approaches are commonly employed: direct shooting [32, 33], and collocation [12, 31]. In the shooting method, the equations of motion are integrated using the nodal values of the inputs, usually by linearly interpolating the inputs between nodes. This is the approach taken here.

The objective in this problem is to find the best trajectories that will place a 50,000 kg payload to Earth orbit while maximizing both the perigee and the apogee distance. The particular launch vehicle used in this example is taken from [26]. The flight mechanics models of the Advanced Launch Vehicle are summarized in Appendix A. The vehicle, to be flown using trimmed aerodynamics until staging and untrimmed aerodynamics after staging, has two control inputs: angle of attack, and velocity roll angle vales specified at the nodes. Nine nodes are used for the period until staging occurs and five nodes from staging to burnout. This problem is clearly much more difficult than the previous problems, since it has many more optimization parameters: 14 angle of attack values, 14 velocity-roll-angle values.

To carry out the optimization, a Fortran program is written to simulate the flight mechanics equations using the inputs given. The program is verified by repeating the single objective optimization problem presented in [26]. The aim of the single objective optimization problem as defined in the named reference was to place the rocket to its final circular orbit at 148.16 km altitude in minimum time, such that any unused fuel may be counted toward payload.

The vehicle is also launched from the −80.54° longitude and 28.5° latitude vertically [26] in the multiobjective optimization problem addressed here. However, it is required to achieve an orbit with an orbital inclination of 28.5° while maximizing both perigee and apogee distances.

$$\text{Maximize} \quad \begin{cases} f_1 = r_{\text{apogee}}, \\ f_2 = r_{\text{perigee}}. \end{cases} \tag{33}$$

The perigee and apogee distances are functions of orbit velocity, altitude and flight path angle at orbit insertion. One may also use equality constraints on orbital inclination, i_f. However, it is found out that the better results shall cause the orbital inclination to approach 28.5° (launch site latitude) since any inclination maneuver requires additional energy causing a reduction in the objective function values. For this reason a rather liberal inequality constraint is imposed on inclination:

$$28.5^\circ \leq i_f \leq 50^\circ. \tag{34}$$

The angles of attack and velocity roll angles were selected from the ±10° range. The optimization is carried out to maximize nine linear FFs (Eq. 26). The results of the optimization, after 100k iterations, are presented in Fig. 11. The figure contains 82 non-dominated points spreading well throughout the front. This problem required about 961 s to run on a 1.7 GHz, 512 MByte, P4 computer with Windows XP operating system.

4.6 Parametric study

As may be observed from the above discussion, MC-MOSA algorithm has mainly two basic parameters: number of FFs used, and the number of iterations or function evaluations permitted (FEN). In addition, the FFs type (i.e., linear or elliptic) has also an important bearing on the front. In this section by the help of two test problems, TP1 and TP2, presented above, the effect of these parameters in obtaining Pareto front is examined. For this purpose quality metrics are used. Many different metrics are proposed in the literature [17, 37, 38]. The problems associated with these metrics in evaluating the quality of the front have also been discussed in the literature [17, 37]. The quality metrics used in this study are taken from [38] and also summarized in Appendix B.

The vales of the metrics presented in Tables 1–5 are the average of 30 separate optimization runs with indicted number of FFs and FEN. The standard deviation of the data is also presented inside the parenthesis. For each test problem, two separate tables are constructed corresponding to linear and elliptic FFs results.

Table 1 presents the metrics obtained for TP1 using linear FFs. It may be observed that as the number of FFs increase, the number non-dominated solutions (NDpts) obtained also increase. For example at 1,000 FEN (i.e., 1k), on the average 12.3 NDpts are obtained using 3FFs. It has increased to 25.5 NDpts with 27 FFs. Similarly when FEN is 20k, these averages for 3FFs and 81 FFs are 17.3 and 64.8 NDPts, respectively. In addition, the standard deviations, although increase with increasing FFs and NDpts, as expected, its ratio to the average values in fact decreases. The other metrics described in Appendix B, shows the same trend as well. For example, as FFs increase, the hyperarea difference (HD) decreases in all FEN categories, indicating a front closer to the actual front. Similarly, accuracy (A) and OS increases with increasing FFs as well. The cluster value for the step size of 1/50 (i.e., 50×50 grid in the space of normalized objectives) $CL_{1/50}$ is also given. Note that his value increases with increasing FFs. This is due to the fact that there are more points on the front, and

Table 1 TP1 results obtained using linear FFs

FEN	FFs	NDpts	HD	A	OS	$CL_{1/50}$	$CO_{1/50}$
1k	3	12.3(2.4)	0.194(0.016)	17.1(5.1)	0.599(0.122)	1.04(0.07)	11.9(2.5)
	9	21.0(3.2)	0.164(0.008)	25.6(6.1)	0.588(0.134)	1.11(0.06)	19.0(2.4)
	27	25.5(3.2)	0.159(0.007)	27.5(6.4)	0.593(0.145)	1.13(0.08)	22.5(2.4)
5k	3	16.4(2.6)	0.184(0.011)	20.2(6.0)	0.702(0.117)	1.11(0.11)	14.7(2.0)
	9	30.1(3.9)	0.152(0.005)	35.1(8.7)	0.660(0.124)	1.18(0.09)	25.6(3.1)
	27	40.5(4.9)	0.147(0.004)	42.4(8.5)	0.661(0.090)	1.24(0.09)	32.6(3.2)
	81	44.7(4.7)	0.148(0.004)	52.1(13.5)	0.733(0.126)	1.27(0.07)	35.2(3.2)
10k	3	15.7(3.1)	0.183(0.015)	19.3(5.2)	0.682(0.116)	1.08(0.07)	14.5(3.0)
	9	33.9(3.8)	0.150(0.005)	43.2(10.5)	0.726(0.112)	1.19(0.07)	28.6(3.0)
	27	49.4(5.5)	0.145(0.004)	56.3(19.1)	0.715(0.102)	1.33(0.09)	37.2(2.8)
	81	53.8(4.3)	0.145(0.005)	60.2(20.0)	0.730(0.120)	1.37(0.11)	39.4(2.4)
20k	3	17.3(2.5)	0.188(0.014)	18.2(5.5)	0.752(0.097)	1.15(0.09)	15.0(2.1)
	9	37.3(3.6)	0.149(0.005)	45.6(9.8)	0.762(0.112)	1.26(0.08)	29.6(2.7)
	27	57.3(4.0)	0.144(0.004)	62.0(14.6)	0.768(0.087)	1.42(0.09)	40.6(3.5)
	81	64.8(6.5)	0.142(0.004)	63.3(16.2)	0.732(0.098)	1.47(0.10)	44.2(2.7)

Average of 30 separate runs are presented, while those numbers inside parenthesis are standard deviations

Table 2 TP1 results obtained using elliptic FFs

FEN	FFs	NDpts	HD	A	OS	$CL_{1/50}$	$CO_{1/50}$
1k	3	9.8(2.3)	0.299(0.027)	10.2(3.9)	0.781(0.11)	1.15(0.19)	8.6(2.1)
	9	15.2(2.2)	0.213(0.011)	18.6(3.8)	0.685(0.156)	1.16(0.22)	13.4(2.7)
	27	29.5(3.4)	0.188(0.007)	33.0(7.1)	0.67(0.104)	1.15(0.08)	25.8(3.0)
5k	3	10.9(2.5)	0.302(0.021)	10.7(4.4)	0.859(0.098)	1.21(0.42)	9.5(2.5)
	9	18.9(2.4)	0.198(0.006)	21.4(3.7)	0.739(0.122)	1.25(0.22)	15.5(2.9)
	27	40.3(4.5)	0.175(0.003)	43.2(6.7)	0.729(0.117)	1.32(0.45)	31.8(4.9)
	81	73.2(4.5)	0.171(0.003)	60.2(9.5)	0.715(0.057)	1.47(0.08)	49.8(3)
10k	3	11.6(2.0)	0.305(0.02)	9.9(3.6)	0.851(0.051)	1.23(0.22)	9.6(2.1)
	9	19.9(2.9)	0.197(0.005)	20.8(3.7)	0.754(0.108)	1.27(0.07)	15.6(2.4)
	27	43.1(4.0)	0.173(0.003)	45.7(6.9)	0.732(0.086)	1.32(0.15)	32.8(3.3)
	81	85.6(5.6)	0.168(0.002)	73.2(12.6)	0.745(0.061)	1.56(0.10)	55.0(3.6)
20k	3	11.8(2.3)	0.301(0.02)	9.4(3.5)	0.845(0.02)	1.28(0.15)	9.4(2.0)
	9	22.0(3.1)	0.195(0.004)	22.5(4.7)	0.779(0.095)	1.29(0.14)	17.2(2.9)
	27	49.8(4.1)	0.171(0.003)	53.0(14.2)	0.790(0.154)	1.70(1.38)	34.7(8.0)
	81	99.1(5.0)	0.166(0.002)	80.8(13.8)	0.759(0.058)	1.70(0.15)	58.8(4.8)

Average of 30 separate runs are presented, while those numbers inside parenthesis are standard deviations

Table 3 TP2 results obtained using linear FFs

FEN	FFs	NDpts	HD	A	OS	$CL_{1/50}$	$CO_{1/50}$
1k	3	11.4(2.7)	0.628(0.068)	5.1(2.1)	0.919(0.04)	1.13(0.13)	10.1(2.2)
	9	14.6(2.5)	0.576(0.033)	7.6(2.4)	0.913(0.047)	1.10(0.08)	13.4(2.1)
	27	16.5(3.1)	0.566(0.033)	7.5(2.2)	0.917(0.06)	1.11(0.09)	14.8(2.8)
5k	3	14.1(2.2)	0.595(0.043)	5.8(2.3)	0.966(0.022)	1.16(0.12)	12.2(2.1)
	9	19.5(3.5)	0.58(0.037)	7.2(2.8)	0.950(0.03)	1.24(0.13)	15.8(2.5)
	27	24.7(4)	0.565(0.047)	8.4(3.9)	0.960(0.024)	1.28(0.12)	19.5(3.3)
10k	3	16.5(3.5)	0.608(0.06)	5.6(2.4)	0.965(0.018)	1.30(0.16)	12.9(2.9)
	9	24.6(4.8)	0.583(0.058)	7.1(2.8)	0.969(0.018)	1.40(0.18)	17.8(3.7)
	27	28.5(5)	0.568(0.037)	7.4(2.9)	0.968(0.023)	1.39(0.14)	20.5(3.0)
	81	32.0(4.5)	0.547(0.025)	9.5(2.8)	0.969(0.017)	1.48(0.16)	21.8(3.7)
20k	3	16.6(4.3)	0.629(0.067)	4.9(2.5)	0.968(0.014)	1.36(0.18)	12.4(3.4)
	9	25.6(3.7)	0.586(0.037)	6.7(2.7)	0.969(0.012)	1.43(0.14)	18.1(2.7)
	27	33.0(3.7)	0.576(0.049)	7.3(2.6)	0.971(0.011)	1.58(0.17)	21.1(3.2)
	81	38.3(4.6)	0.552(0.027)	8.8(3.1)	0.969(0.013)	1.65(0.15)	23.2(2.2)

Average of 30 separate runs are presented, while those numbers inside parenthesis are standard deviations

the results of the metric are not conclusive. It is clear that as FFs increase, the solutions occupy many more cells, indicating a well spread front (i.e., $CO_{1/50}$ increases). Although increasing FFs increase the computation cost; it is not as costly as increasing the FEN, in many real life optimization problems such as the trajectory optimization problem, given above.

On the other hand, increase in FEN increases NDpts, decreases HD, while A, OS also increases. For example, with 27FFs, at 1k the HD is 0.159, while it is reduced to 0.144 at 20k. $CO_{1/50}$ also increases from 22.5 at 1k to 40.6 at 20k, when 27FFs are employed. Similar improvements are observable for A and OS. Consequently,

Table 4 TP2 results obtained using elliptic FFs

FEN	FFs	NDpts	HD	A	OS	$CL_{1/50}$	$CO_{1/50}$
1k	3	8.2(2.2)	0.643(0.057)	5.2(2.7)	0.894(0.138)	1.11(0.08)	7.5(2.2)
	9	21.1(2.9)	0.521(0.009)	14.9(2.3)	0.932(0.050)	1.13(0.0)	18.6(2.3)
	27	43.1(4.5)	0.492(0.005)	33.5(4.7)	0.92(0.057)	1.14(0.05)	37.6(3.8)
	81	58.1(7.3)	0.488(0.004)	43.5(7.7)	0.918(0.042)	1.25(0.09)	46.3(4.5)
5k	3	10.2(2.4)	0.651(0.034)	4.7(1.8)	0.982(0.026)	1.21(0.14)	8.5(1.9)
	9	30.1(4.3)	0.515(0.006)	15.2(2.5)	0.983(0.023)	1.30(0.11)	23.3(3.4)
	27	73.1(73.1)	0.483(0.001)	48.6(4.1)	0.956(0.029)	1.44(0.10)	49.8(3.8)
	81	134.4(6.2)	0.476(0.001)	99.5(9.9)	0.966(0.026)	1.83(0.16)	74.0(6.2)
10k	3	10.5(2.9)	0.649(0.041)	4.6(1.7)	0.986(0.024)	1.33(0.23)	8.1(2.3)
	9	34.0(3.7)	0.516(0.005)	14.9(1.9)	0.980(0.043)	1.47(0.13)	23.2(2.5)
	27	82.9(5.9)	0.481(0.001)	51.2(4.8)	0.962(0.020)	1.57(0.11)	52.4(4.6)
	81	170.8(7.3)	0.474(0.0)	123.3(6.8)	0.965(0.019)	2.18(0.22)	79.0(6.9)
20k	3	11.8(2.1)	0.651(0.041)	4.2(1.3)	0.991(0.015)	1.52(0.26)	7.9(1.6)
	9	39.6(5.1)	0.512(0.006)	15.6(2.6)	0.986(0.039)	1.60(0.18)	24.9(3.4)
	27	96.8(6.8)	0.481(0.001)	52(4.4)	0.969(0.023)	1.79(0.10)	54.2(4.3)
	81	211.8(10.4)	0.473(0.0)	145.1(9.5)	0.970(0.016)	2.60(0.30)	82.4(8.2)

Average of 30 separate runs are presented, while those numbers inside parenthesis are standard deviations

Table 5 Comparison of NSGA-II results with MCMOSA for TP1

	NSGA-II	MC-MOSA		
	100 members	243 Elliptic FFs		243 Linear FFs
FEN	10k	5k	10k	10k
NDpts	95.9(4.3)	87.8(5.2)	111.1(7.5)	55.3(5.4)
HD	0.140(0.005)	0.171(0.003)	0.167(0.003)	0.150(0.001)
A	62.1(7.6)	66.1(15.9)	79.7(18.9)	54.2(14.8)
OS	0.635(0.040)	0.730(0.072)	0.746(0.073)	0.704(0.100)
$CL_{1/50}$	1.47(0.06)	1.70(0.12)	1.93(0.12)	1.40(0.1)
$CL_{1/150}$	1.10(0.04)	1.12(0.05)	1.18(0.06)	1.10(0)
$CO_{1/50}$	65.2(3.1)	51.97(3.78)	57.73(4.07)	39.9(3.7)
$CO_{1/150}$	87.0(5.0)	78.3(4.86)	94.43(7.07)	51.5(5.2)

Results are the average of 30 separate runs. Standard deviations are given inside parentheses

the increase in FEN also gives NDpts closer to the actual Pareto front, with higher accuracy and better more uniform spread.

TP1 is also solved using elliptic FFs as well (Table 2). In this case, ellipse centers are placed on a quarter circle of radius 1.5 in the normalized objective space (Fig. 3). From the table it may be observed that when too many FFs are used (i.e., 81), more NDpts are obtained with elliptic FFs, than linear FFs. However, opposite is true when fewer FFs are used. Similarly the values of metrics are better when few linear FFs are used than elliptic FFs. For example with 27 FFs at 20k iterations linear FFs give 57.3 NDpts, where elliptic gives only 49.8 FFs. However, with 81 FFs, and after 20k iterations, linear FFs this time give only 64.8 NDpts, while elliptic FFs give 99.1 NDpts. Same trend is observable in the accuracy metric (A), and cells occupied ($CO_{1/50}$). Except HD, the same trend is observable with the other metrics. When HD is considered, the linear FFs are observed to be more successful. This may be due to the fact that the front is already quite convex.

The study is repeated this time for TP2. As discussed before the boundary of this test problem has a nice convex part and almost straight-line part. For this reason, the elliptic FFs are expected to be more successful in this problem. The results of the study are presented in Tables 3 and 4. First it may be observed that elliptic FFs are more successful than linear FFs in all categories if more than 3FFs are used. On the other hand, more non-dominated solutions are obtained as the number of FFs increase. In addition, they are closer to the actual front with improved accuracy (A) as well as better and more uniform spread (greater OS and $CO_{1/50}$ value).

4.7 Comparison with NSGA II

In this section, MC-MOSA algorithm is compared with the NSGA-II algorithm [7]. The program is downloaded from the related web site and run for the first three test problems (TP1, TP2, and TP3) thirty times, using a population of 100 evaluating for 100 generations. This corresponds to 10k function evaluations. In the last problem, a 100-member population is evolved for 50 generations (i.e., 5k function evaluations). During the runs, different seeds for the random number generator and different parameters are selected from the range recommended by the program. Then, average values and standard deviations of the metrics are calculated.

Table 5 compares the results obtained with NSGA-II and MC-MOSA for TP1. For MC-MOSA 243 linear as well as elliptic FFs are employed while the results presented are the averages of the 30 separate runs as before. It may be observed from the table that MC-MOSA with elliptic FFs gives more points than NSGA-II at 10k. The HD value of NSGA-II (0.14) is better than MC-MOSA results with elliptic FFs, while linear FFs gives a closer value of 0.15. Overall spread and accuracy values of the MC-MOSA obtained set with elliptic FFs are better than NSGA-II results. NSGA-II occupies more cells when step size of 1/50 is used. With a step size of 1/150, MC-MOSA with elliptic FFs, performs better at 10k, since it has more non-dominated points. Overall it may be concluded that in this numerical experiment, with lower HD value and higher CO values, NSGA-II performed better than MC-MOSA, although MC-MOSA runs averaged better values in the remaining metrics.

Table 6 gives the quality metrics obtained for TP2 using NSGA-II and MC-MOSA algorithms. In these tables MC-MOSA results with only elliptic FFs are presented. It may be observed from the table that MC-MOSA results are much superior than NSGA-II even with 5k iterations. Thus, MC-MOSA found more non-dominated points, had a smaller HD, better accuracy (A) and spread (OS), while occupying more cells in both grid sizes.

Table 6 Comparison of NSGA-II results with MCMOSA for TP2

	NSGA-II 100 members	MC-MOSA 243 Elliptic FFs	
FEN	10k	5k	10k
NDpts	96.1(2.5)	166(9.4)	236.7(11.7)
HD	0.478(0.004)	0.475(0.001)	0.473(0)
A	75.9(8.0)	119.7(11.1)	166.4(13.8)
OS	0.944(0.044)	0.953(0.028)	0.968(0.02)
$CL_{1/50}$	1.47(0.16)	2.20(0.23)	2.89(0.30)
$CL_{1/150}$	1.06(0.03)	1.23(0.05)	1.40(0.05)
$CO_{1/50}$	66.1(6.7)	76.0(6.8)	82.7(8.3)
$CO_{1/150}$	90.4(3.7)	134.5(171.4)	168.6(217.4)

Results are the average of 30 separate runs. Standard deviations are given inside parentheses

Table 7 Comparison of NSGA-II results with MCMOSA for TP3

	NSGA-II 100 members	MC-MOSA 250 elliptic FFs
FEN	10k	10k
NDpts	73.9(16.1)	65.4(6.4)
HD	0.39(0.012)	0.386(0.004)
A	37.9(27.5)	66.1(15.4)
OS	0.719(0.18)	0.875(0.028)
$CL_{1/25}$	3.29(1.04)	2.2(0.19)
$CL_{1/100}$	1.43(0.37)	1.14(0.05)
$CO_{1/25}$	23.9(7.1)	29.7(1.7)
$CO_{1/100}$	55.1(17.3)	57.4(5.3)

Results are the average of 30 separate runs. Standard deviations are given inside parentheses

Table 8 Comparison of NSGA-II results with MCMOSA for TP4

	NSGA-II 100 members	MC-MOSA 496 Ellipsoidal FFs
FEN	5k	5k
NDpts	96.4(3.6)	118.9(12.9)
HD	0.386(0.031)	0.425(0.01)
A	88.3(21.1)	91.2(26.5)
OS	0.802(0.238)	0.853(0.053)
$CL_{1/10}$	1.55(0.27)	1.41(0.08)
$CL_{1/50}$	1.06(0.04)	1.02(0.01)
$CO_{1/10}$	63.8(8.8)	84.1(7)
$CO_{1/50}$	90.8(4.1)	116.4(12.6)

Results are the average of 30 separate runs. Standard deviations are given inside parentheses

Table 7 presents the results for the third test problem (TP3). In this problem, the front obtained using MC-MOSA resulted in better HD, A, OS, CO metrics than NSGA-II. Although NSGA-II gave more non-dominated points, its CO values are inferior to those of MC-MOSA. Thus, MC-MOSA has occupied more cells both in 25×25 grids and 100×100 grids.

The comparison is also carried out for the three objective optimization problem, TP4. In two objective case HD, A, OS metrics represent areas, where CL and CO metrics are related to the occupation of rectangular areas. When there are three objectives, we have volumes instead of areas. Consequently, proper algorithms are developed to calculate the volume below the points in the set for the HD metric, volumes bounded by neighboring points for the A metric, and volume of the extreme rectangular prism or the OS metric. Similarly, now the volumes of rectangular prisms that contain a solution are counted to obtain the CO or CL metrics as well.

Since TP4 is a rather easy problem, it is decided to solve a four optimization variable problem as stated in Eq. 29, instead of the two optimization variable solution presented in Fig. 15. Thus, all four cross-sectional areas are treated as an optimization variable. To get a meaningful difference between the metrics, a 5k function evaluation problem is solved for MC-MOSA. Similarly, in NSGA-II a 100-member population is iterated for 50 generations. The results of the optimization are presented in Table 8. It should be noted that in one experiment, NSGA-II performed very poorly. For fairness, this data is replaced by the data from an additional run. The results given in Table 8 show that NSGA-II has given a better HD metric in the average than MC-MOSA, however, with a large standard deviation. In the remaining metrics, MC-MOSA is clearly the winner.

5 Conclusion

In this manuscript, a new multiobjective simulated annealing algorithm is presented. Algorithm is based on the single objective simulated annealing algorithm, Hide-and-Seek [1, 25]. The algorithm uses many fitness functions constructed from the objectives to approximate the Pareto-front closely. Each fitness function has a separate temperature parameter and cooled separately whenever an improvement to that fitness function is encountered. Unlike other SA or EA based multiobjective algorithms the proposed algorithm does not have problem dependent parameters to adjust.

The effectiveness of the algorithm is shown through five test problems. Also presented is a parametric study to show the effect of the number of fitness functions employed and number of iterations carried out on the success of the algorithm. The effect of FFs on the function type (linear or elliptic) is also examined and discussed. It is shown that more fitness functions give more points close to the actual Pareto front. Similarly as the permitted number of iterations increase, not only more non-dominated points are obtained these points are located closer to the actual Pareto front. However, it should be remembered that using more fitness functions is cheaper than increasing the number of function evaluations in multidisciplinary design problems. Additionally, while the linear FFs are good for highly convex fronts, elliptic FFs are observed to be better for non-convex, or slightly non-convex fronts. Note that there is no reason to use only one type of fitness functions. A population containing many different types of fitness functions may be employed together for the generation of non-convex fronts.

Through numerical experiments, MC-MOSA algorithm is also compared to the well-known NSGA-II algorithm [7] through quality metrics. It is observed that in the first problem NSGA-II performed better than MC-MOSA. In the second and third test problems, MC-MOSA was better than NSGA-II in all quality metrics considered. Finally, in the three-objective problem, NSGA-II and MC-MOSA displayed similar performances. Obviously, a more rigorous comparison together with proper statistical analysis is required before declaring any superiority of one algorithm over other. However, the numerical experiments suggest that MC-MOSA performs as well as NSGA-II on the test functions under consideration, although a rigorous statistical analysis would be required to conclude with confidence.

Acknowledgments The authors would like to acknowledge the anonymous referees for their valuable comments that helped improve the quality of this manuscript.

Appendix A: advanced launch vehicle model [26]

Advanced Launch Vehicle flight mechanics equations, written with respect to an Earth centered, Earth fixed frame are given below:

$$\dot{\lambda} = \frac{V\cos\gamma\cos\psi}{r\cos\tau},$$
$$\dot{\tau} = \frac{V\cos\gamma\sin\psi}{r},$$
$$\dot{h} = V\sin\gamma,$$
$$\dot{V} = \frac{1}{m}(T\cos(\alpha+\delta) - D - mg\sin\gamma) + r\omega^2\cos\tau(\cos\tau\sin\gamma - \sin\tau\cos\gamma\sin\psi),$$

$$\begin{aligned}\dot{\gamma} &= \frac{1}{mV}[(T\sin(\alpha+\delta)+L)\cos\mu - mg\cos\gamma] + \frac{V\cos\gamma}{r}\\ &\quad + 2\omega\cos\tau\cos\psi + \frac{r\omega^2}{V}\cos\tau(\cos\tau\cos\gamma + \sin\tau\sin\gamma\sin\psi),\\ \dot{\psi} &= -\frac{1}{mV\cos\gamma}(T\sin(\alpha+\delta)+L)\sin\mu - \frac{V}{r}\tan\tau\cos\gamma\cos\psi\\ &\quad + 2\omega(\cos\tau\tan\gamma\sin\psi - \sin\tau) - \frac{r\omega^2}{V\cos\gamma}\cos\tau\sin\tau\cos\psi.\end{aligned} \tag{35}$$

In the above equations, λ is the longitude, τ the latitude, h the altitude above mean sea level, V the total velocity, γ the flight path angle, ψ the heading angle, m the mass, r the distance from the center of the Earth to the vehicle mass center, ω the angular velocity of the Earth, and D, L, T, are the drag, lift, thrust forces, respectively. The inputs are angle of attack, α, and velocity roll angle, μ. The lift and drag coefficients are interpolated from the tables given according to the angle of attack and Mach number, for two configurations: with and without boosters. The launch vehicle has ten identical thrusters, each producing the same thrust of 2,580 kN at vacuum, while all of them are ignited at launch. Seven of these thrusters are boosters and separated from the vehicle immediately after the depletion of the fuel, 153.5 s after ignition. The remaining three thrusters of the main body are active for 365.5 s from ignition. The change in thrust value according to the ambient pressure is also taken into account: $T' = T'_{\text{vac}} - pA'_{\text{e}}$, where T'_{vac} is the vacuum trust, p the ambient pressure, and A'_{e} is the nozzle exhaust area of each thruster. The total mass of the launcher is 1,584,532 kg, including a 59,874 kg nominal payload. Due to fuel consumption, the mass of the vehicle is reduced with time according to the following formula.

$$\dot{m} = -\frac{1}{I_{\text{sp}}g_{\text{s}}}T_{\text{vac}}, \tag{36}$$

where, T_{vac}, total thrust at vacuum of the active thrusters, I_{sp} is the specific impulse and g_{s} is the gravitational attraction at sea level. Since in the above flight mechanics equations, the angle of attack is an input, the trimmed flight mechanics equations are used until staging. The required gimbal angle for trim is found from:

$$\delta = -\frac{M_{\text{A}}}{Tl_t}, \tag{37}$$

where, M_{A} is the aerodynamic pitching moment and l_t is the distance from the center of mass to the exit plane of the engines. The aerodynamic coefficients are given with respect to the center of mass location at launch. Consequently, proper calculations are carried out to find the pitch moment at the instant required. After staging untrimmed aerodynamics is used ($M_{\text{A}} \approx 0$, $\delta = 0$). Gravity model that uses inverse square law, and an exponential atmosphere models are also employed.

The above flight vehicle equations are singular when the vehicle velocity is zero as well as when the flight path angle is 90°. To avoid singularity, the integration of the equations is started three seconds after the ignition, when the vehicle attains sufficient velocity and at this instant flight path angle is taken to be 89°. The inertial velocity of the vehicle and the orbital inclination are found from the following equations:

$$V_i = [V^2 + 2Vr\omega\cos\gamma\cos\psi\cos\tau + (r\omega\cos\tau)^2]^{1/2}, \tag{38}$$

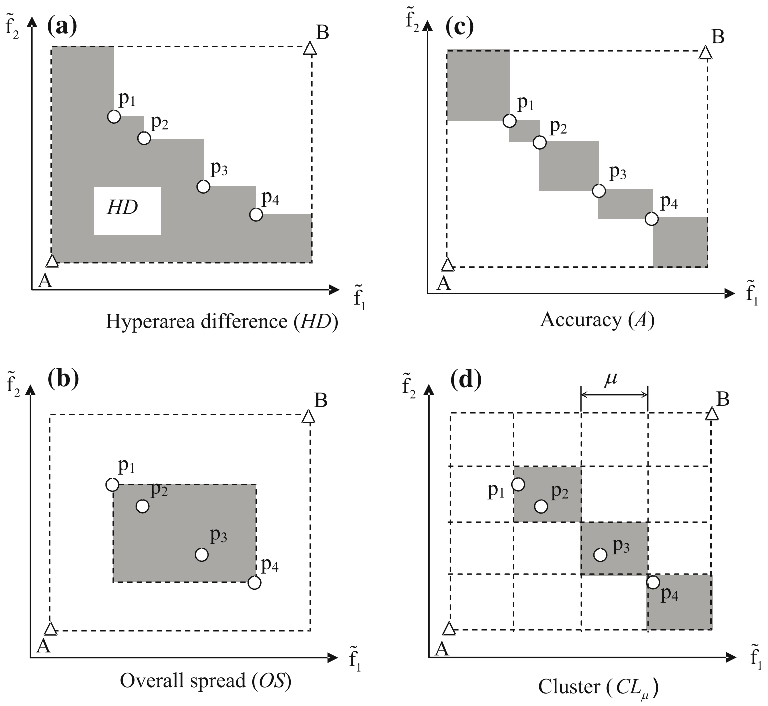


Fig. 17 Geometric description of the metrics used in the study [38], with normalized objectives

$$\cos i = \frac{\cos\tau(V\cos\gamma\cos\psi + r\omega\cos\tau)}{[V^2\cos^2\gamma + 2Vr\omega\cos\gamma\cos\psi\cos\tau + (r\omega\cos\tau)^2]^{1/2}}. \tag{39}$$

Appendix B: quality metrics

In this manuscript, the quality assessment of the frontier obtained through multi-objective optimization are conducted using four metrics proposed in [38] are used. They are: the hyper-area difference (HD), overall Pareto spread (OS), accuracy of the observed Pareto frontier (AC), cluster (CL).

The hyperarea difference is the area below the Pareto frontier, as shown in Fig. 17a. Points A and B defines the bounding box around the Pareto front. Normally, HD shall be calculated using normalized objectives. Then, in general, it is claimed that the smaller the HD metric is, the better the observed Pareto solution set [38]. Overall Pareto spread (OS) is the area of the maximum rectangle constructed using the two extremes of the Pareto front (p_1 and p_2) as shown in Fig. 17b. Again A solution set with the largest OS value is generally an indication that a particular front has spread to the extreme ends of the Pareto front, and consequently, it is comparatively better than a front with a smaller value. Accuracy (A) is a measure how smooth the observed front is. To obtain this metric, areas of the small rectangles constructed from neighboring solutions are summed up (Fig. 17c) to obtain a total area. The metric is defined as the inverse of this total area. If the solution set contains all the actual

Pareto solutions (i.e., a continuous Pareto frontier), then the total area will be zero, causing the AC metric to be infinite. Thus, a solution set with a large AC value is better than the one with a smaller AC value. It is desirable to have the solutions spread uniformly along the front. Clustering occurs, when too many solutions are found at certain parts of the front, while other parts are empty. To obtain the CL_μ metric, the whole domain normalized objectives is divided into square grids of size μ. Then, those rectangles occupied with a non-dominated solution are counted. The total number of non-dominated solutions in the set is divided to the number of occupied rectangles. Ideally, to have a good spread, each rectangle shall be occupied by a single solution giving a CL_μ metric equal to one. For example, in Fig. 17d there are four solutions in the front, while only three grids are occupied (i.e., $CL_\mu = 1.25$). Similarly, of the two solution sets having almost equal number of solutions, the one with a smaller CL_μ metric shall be preferred. However, this comparison may not be meaningful if each set contains very different number of solutions. For this reason, in this manuscript the number of cells occupied by a non-dominated solution (CO_μ) is also used as a metric. In this case, the greater the CO_μ value, the better the solution is.

References

1. Belisle, C.J.P., Romeijin, H.E., Smith, R.L.: Hide-and-Seek a Simulated Annealing Algorithm for Global Optimization. Technical Report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, No: 90–25 (1990)
2. Chipperfield, A., Fleming, P.J.: Multiobjective gas turbine engine controller design using genetic algorithms. IEEE Trans. Ind. Electron. **43**(5), 1–5 (1996)
3. Coello, C.A.C.: Guest editorial: special issue on evolutionary multiobjective optimization. IEEE Trans. Evol. Comput. **7**(2), 97–99 (2003)
4. Corona, A., Marchesi, M., Martini, C., Ridella, S.: Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. ACM Trans. Math. Softw. **13**(3), 262–280 (1987)
5. Czyżak, P., Jaszkiewicz, A.: Pareto simulated annealing - a metaheuristic technique for multiple objective combinatorial optimization. J. Multi-Criteria Decision Anal. **7**, 34–47 (1998)
6. Dakev, N.V., Whidborne, J.F., Chipperfield, A.J., Fleming, P.J.: Evolutionary H-infinity design of an electromagnetic suspension control system for a maglev vehicle. Proc. Ins. Mech. Eng. Part I J. Sys. Control Eng. **211**, 345–355 (1997)
7. Deb, K.: Multiobjective Optimization Using Evolutionary Algorithms. Wiley, New York (2001)
8. Edgeworth, F.Y.: Mathematical Psychics; an Essay on the Application of Mathematics to Moral Sciences. C.K. Paul and Co., London (Reprints of Economic Classics, A.M. Kelley, New York, 1961) (1891)
9. Haas, O.C.L., Burnham, K.J., Mills, J.A.: On improving physical selectivity in the treatment of cancer: a systems modeling and optimization approach. Control Eng. Pract. **5**(12), 1739–1745 (1997)
10. Hajek, B.: Cooling schedules for optimal annealing. Math. Oper. Res. **13**(2), 311–329 (1988)
11. Hajela, P.: Nongradient methods in multidisciplinary design optimization—status and potential. J. Aircraft. **36**(1), 255–265 (1999)
12. Hargraves, C.R., Paris, S.W.: Direct trajectory optimization using non-linear programming and collocation. J. Guidance Control Dyn. **10**(4), 339–342 (1987)
13. Hartmann, J.W., Coverstone-Caroll, V.L., Williams, S.N.: Optimal interplanetary trajectories via a Pareto genetic algorithm. J. Astron. Sci. **46**(3), 267–282 (1998)
14. Hull, D.G.: Conversion of optimal control problems into parameter optimization problems. J. Guidance, Control Dyn. **20**(1), 57–60 (1997)
15. Karsli, G.: Simulated annealing for the generation of Pareto fronts with aerospace applications. MS Thesis, Middle East Technical University Libraries, Ankara, Turkey (2004)
16. Kirkpatrick, S., Gelatt, C.D., Vechi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
17. Knowles, J., Corne, D.: On metrics comparing non-dominated sets. In: Proceedings of the 2002 Congress on Evolutionary Computation, pp. 711–716. IEEE Press, Piscataway, NJ (2002)
18. Kubotani, H., Yoshimura, K.: Performance evaluation of acceptance probability functions for multiobjective SA. Comput. Oper. Res. **30**, 427–442 (2003)

19. Lučić, P., Teodorović, D.: Simulated annealing for the multiobjective aircrew rostering problem. Trans. Res. Part A **33**, 19–45 (1999)
20. Lu, P., Khan, M.A.: Nonsmooth trajectory optimization: An approach using continuous simulated annealing. J. Guidance Control Dyn. **17**(4), 685–691 (1994)
21. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
22. Messac, A., Mattson, C.A.: Normal constraint method with guarantee of even representation of complete pareto frontier. AIAA J **42**(10), 2101–2111 (2004)
23. Pareto, V.: In: Schwier, A.S., Page, A.N. (eds.) Manual of Political Economy (translated from 1927 French edition by Ann S. Schwier) Augustus M. Kelley Publishers, New York (1971)
24. Rajesh, J.K., Gupta, S.K., Rangaiah, G.P., Ray, A.K.: Multiobjective optimization of industrial hydrogen plants. Chem. Eng. Sci. **56**, 999–1010 (2001)
25. Romeijn, H.E., Smith, R.L.: Simulated annealing for constrained global optimization. J. Glob. Optim. **5**, 101–126 (1994)
26. Shaver, D.A., Hull, D.G.: Advanced launch system trajectory optimization using suboptimal control. AIAA Paper 90–3413 (1990)
27. Siarry, P., Berthian, G., Durbin, F., Hamesy, J.: Enhanced simulated annealing for globally minimizing functions of many continuous variables. ACM Trans. Math. Softw. **23**(2), 209–228 (1997)
28. Suman, B.: Simulated annealing-based multiobjective algorithms and their application for system reliability. Eng. Optim. **35**(4), 391–416 (2003)
29. Suppaptnarm, A., Seffen, K.A., Parks, G.T., Clarkson, P.J.: Simulated annealing: an alternative approach to true multiobjective optimization. Eng. Optim. **33**(1), 59–85 (2000)
30. Teghem, J., Tuyttens, D., Ulungu, E.L.: An interactive heuristic method for multi-objective combinatorial optimization. Comput Oper Res **27**, 621–634 (2000)
31. Tekinalp, O., Arslan, E.M.: Optimal mid-course trajectories for precision guided munitions with collocation and nonlinear programming. In: Proceedings of the 20th Congress of the International Council of Aeronautical Sciences, vol. 1, pp. 593–600. AIAA, Reston, VA (1996)
32. Tekinalp, O., Bingol, M.: Simulated annealing for missile optimization: Developing method and formulation techniques. J. Guidance, Control Dyn. **27**(4), 616–626 (2004)
33. Tekinalp, O., Utalay, S.: Simulated annealing for missile trajectory planning and multidisciplinary missile design optimization. AIAA Paper No: 2000-0684 (2000)
34. Ulungu, E.L., Teghem, J., Fortmeps, P.H., Tuyttnes, D.: MOSA method: A Tool for solving multiobjective combinatorial optimization problems. J. Multi-Criteria Decision Anal. **8**, 221–236 (1999)
35. Vanderbilt, D., Lougie, S.G.: A Monte Carlo simulated annealing approach to optimization over continuous variables. J. Comput. Phys. **56**, 259–271 (1984)
36. Whidborne, J.F., Gu, D.-W., Postlethwaite, I.: Simulated annealing for multi-objective control system design. IEE Proc. Control Theory Appl. **144**(6), 582–588 (1997)
37. Zitzler, E.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–137 (2003)
38. Wu, J., Azarm, S.: Metrics for quality assesment of a multiobjective design optimization solution Set. ASME J. Mech. Des. **123**, 18–25 (2001)