Struct Multidisc Optim (2018) 57:509–533
DOI 10.1007/s00158-017-1764-7

RESEARCH PAPER

A new PSO-based algorithm for multi-objective optimization with continuous and discrete design variables

Vahid Mokarram[1] · **Mohammad Reza Banan**[1]

Received: 20 August 2016 / Revised: 31 May 2017 / Accepted: 10 July 2017 / Published online: 26 July 2017

Abstract This paper presents a new multi-objective optimization algorithm called FC-MOPSO for optimal design of engineering problems with a small number of function evaluations. The proposed algorithm expands the main idea of the single-objective particle swarm optimization (PSO) algorithm to deal with constrained and unconstrained multi-objective problems (MOPs). FC-MOPSO employs an effective procedure in selection of the leader for each particle to ensure both diversity and fast convergence. Fifteen benchmark problems with continuous design variables are used to validate the performance of the proposed algorithm. Finally, a modified version of FC-MOPSO is introduced for handling discrete optimization problems. Its performance is demonstrated by optimizing five space truss structures. It is shown that the FC-MOPSO can effectively find acceptable approximations of Pareto fronts for structural MOPs within very limited number of function evaluations.

Keywords Multi-objective optimization · Structural optimization · Particle swarm optimization

1 Introduction

Most engineering design problems can be regarded as MOPs. Ideally, a designer tends to attain an economical solution which represents the best performance for a particular problem. From a practical point of view, structural optimization is not a single-objective problem (SOP) since in fact the designer needs to consider different objective functions such as minimizing the life cycle cost and maximizing the performance of the structure through minimizing its seismic energy input, maximizing its hysteretic energy dissipation and so on. Moreover, MOPs propose sets of optimal solutions known as Pareto optimal sets instead of one single optimal point. Pareto optimal sets are more desirable since decision makers can access a variety of optimal scenarios and choose the one which best suits the demands of a particular project. In the last few decades, evolutionary algorithms (EAs) have attracted considerable attention of researchers. Since EAs eliminate the differentiability requirement for objective functions in classical methods, they can be applied to discrete and continuous optimization problems. Most EAs start the search with random initial points in the search space which removes the difficulty of finding an appropriate initial point. Furthermore, EAs can be applied to highly nonlinear problems. These characteristics make EAs a suitable choice for engineering optimization problems including structural optimization problems. However, in most structural optimization problems each function evaluation may entail multiple structural analyses. Hence, function evaluations can be such costly that make optimization process impractical for real-world problems especially when nonlinear time history analysis must be conducted. As an example of such time consuming processes the reader may refer to the GA-based optimal design of the 3-story building presented by Gong et al. (2013) which lasted 105 h on a desktop computer before converging. Therefore, more emphasis should be put on the number of function evaluations required by EAs for structural problems. One of the disadvantages of EAs is their need for many number of function evaluations before converging to some acceptable approximation

✉ Mohammad Reza Banan
banan@shirazu.ac.ir

Vahid Mokarram
vahid_mokarram@shirazu.ac.ir

[1] Department of Civil and Environmental Engineering, School of Engineering, Shiraz University, Shiraz, Fars, Iran

of the Pareto front. Therefore, designing algorithms for handling the aforementioned drawback is essential. Regarding the performance of MOPs, there are few researches that tried to reduce the number of fitness evaluations. For instance, Chafekar et al. (2005) proposed a method called OEGADO to deal with this problem. They verified their algorithm for a few benchmark problems. However, the method yields acceptable results within a few thousands function evaluations. Eskandari and Geiger (2008) proposed another GA-based method called FastPGA and applied it to a number of unconstrained continuous benchmark problems that could be completed within a few thousands of function evaluations. There are some other techniques that can also be used to reduce the number of function evaluations. These include employing methods for estimating the values of objective functions instead of exact evaluations. An example of such approach can be found in (Davarynejad et al. 2011) where a fuzzy-based technique is utilized for function approximations. It was successfully applied to the unconstrained continuous ZDT problems to obtain acceptable results within 1000 real function evaluations. The reader may refer to (Santana-Quintero et al. 2010) for a more detailed discussion on objective function approximations.

Among EAs, the PSO method is well known for its fast convergence. However, the method was originally proposed for optimization of unconstrained single-objective continuous problems. There exist some extensions of PSO for multiobjective problems in the literature. However, as it is also discussed by Tong et al. (2016), only a few of them are developed for solving problems with discrete and mixed discrete and continuous design variables. In this paper, the original PSO is extended to achieve a fast converging PSO-based algorithm called FC-MOPSO which can solve constrained/unconstrained continuous/discrete as well as mixed continuous and discrete MOPs within a few hundreds of function evaluations. The performance of the method is verified against some well-known EAs.

2 Unconstrained continuous single-objective PSO

The original PSO method was first introduced by Kennedy and Eberhart (1995) for optimization of continuous nonlinear single-objective functions. The method was proposed based on the observation that groups of individuals such as birds work together to improve their performance. Since then PSO has gained much popularity among researchers because of its simplicity and fast convergence. The idea of PSO is to start the search for an optimal solution through some initial random particles with initial random velocities (or initial zero velocities) and to update the velocities by considering two main aspects, cognition learning and social learning. In later versions of PSO as discussed by Clerc and Kennedy (2002), a factor known as constriction factor is applied to the velocity formulation to control explosion of the system. Hence, the basic formulation used in this paper for updating the position of each particle (or the flight stage in PSO) is as follows.

$$\boldsymbol{v}_i^{t+1} = \chi\left[\boldsymbol{v}_i^t + \phi_1 \boldsymbol{r}_1 \circledast (\boldsymbol{pbest}_i - \boldsymbol{x}_i) + \phi_2 \boldsymbol{r}_2 \circledast (\boldsymbol{lbest}_i - \boldsymbol{x}_i)\right] \tag{1}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{2}$$

In this formulation, which has been proposed for the case of single-objective PSO, $\boldsymbol{v}_i^t = \left[v_{i_1}^t, v_{i_2}^t, \ldots, v_{i_s}^t\right]$ is the vector of velocity for the i^{th} particle at the t^{th} generation, $\boldsymbol{x}_i^t = \left[x_{i_1}^t, x_{i_2}^t, \ldots, x_{i_s}^t\right]$ is the position of the i^{th} particle at the t^{th} generation and s is the number of design variables. $\chi = 2\alpha/(\phi_1 + \phi_2 - 2)$ is the constriction factor and ϕ_1 and ϕ_2 are the maximum cognition learning rate and the maximum social learning rate, respectively. $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are vectors consisting of random numbers between zero and one, $\boldsymbol{pbest}_i$ is the best position of i^{th} particle found so far and $\boldsymbol{lbest}_i$ is the local best or the best particle found in the neighborhood of the i^{th} particle. Furthermore, if $\boldsymbol{a} = [a_1, a_2, \ldots, a_n]$ and $\boldsymbol{b} = [b_1, b_2, \ldots, b_n]$ then $\boldsymbol{a} \circledast \boldsymbol{b} \coloneqq [a_1 b_1, a_2 b_2, \ldots, a_n b_n]$.

It is to be noted that many different neighborhood topologies have been used in the literature. In this paper, we will use a random neighborhood topology which will be updated at each generation. Although it might be possible to obtain better results by applying certain neighborhood topologies, the random topology suggests a more general approach which is less problem dependent.

Clerc and Kennedy (2002) have shown that the PSO system is stable if $\phi_1 + \phi_2 > 4$ and $\alpha \in (0, 1)$. In this paper, $\phi_1 = \phi_2 = 2.05$ which has been recommended by Simon (2013) is used. Clerc and Kennedy (2002) also discuss that consistent convergence to local minima can be guaranteed although convergence on global optima cannot be proven except for some special class of functions. Evidently, such theoretical discussions about the convergence of a PSO system is valid only if $t \to \infty$. Furthermore, this theoretical background applies only to the unconstrained continuous single-objective PSO. It would be very difficult to provide similar mathematical discussions for PSO systems which deal with these difficulties: (1) multiobjectiveness (2) discreteness and, (3) constraints. That is why FC-MOPSO will be formulated as an extension of the given PSO variant which has some theoretical background. (1) and (2) represent the main idea of PSO algorithms. However, some modifications must be implemented for handling MOPs. In Section 3, we will propose a modification that preserves the basic rationale of PSO so that FC-MOPSO can inherit the stability characteristics of its ancestor.

By changing the value of α, a tradeoff between the exploration characteristic of a PSO system and its exploitation

characteristic can be obtained. Larger values of α (or larger values of the constriction factor) indicate allowing the swarm to experience larger velocities which results in more exploration. A smaller α, however, emphasizes on more exploitation. In this paper, it is recommended to use the maximum allowable value of α which is practically a value near unity like $\alpha = 0.9$. Because, FC-MOPSO receives much information form the swarm that may result in premature convergence or stagnation if small values of α are used. It is also possible to define a pattern for updating α at each generation. For example, Larger values of α can be used at earlier generations when the swarm is meant to emphasize on exploration while smaller values can be adopted at final generations when the swarm is supposed to converge to the solution. A simple way for accomplishing it, is to define α as a linear or exponential function of t. Using non-constant α values, however, requires parameter tunings which is not desired because it would make the method more problem dependent. Regarding the above discussions, $\alpha = 0.9$ is used throughout this paper.

3 Unconstrained continuous FC-MOPSO

3.1 Definitions

Without loss of generality, any multi-objective problem can be defined as follows.

$$\begin{aligned} &\textit{minimize } f(\mathrm{x}) \\ &\textit{subject to} \\ &g_j(x) \le 0\ ,\ \text{for}\ \ j \in [1,p] \\ &h_j(x) = 0, \text{for } j \in [p+1,q] \end{aligned}$$

where $\boldsymbol{x}$ is the vector of design variables, $f(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}),\ldots,f_k(\boldsymbol{x})]$ is the vector of k objective functions which are to be minimized simultaneously, p is the number of inequality constraints and q is the total number of equality and inequality constraints. Evidently, by putting p and q equal to zero the above definition will represent an unconstrained multi-objective optimization problem. Furthermore, any solution to the above problem should lie in the search space domain, i.e., $\boldsymbol{L} \le \boldsymbol{x}^* \le \boldsymbol{U}$ where $\boldsymbol{L} = [L_1,\ldots,L_s]$ and $\boldsymbol{U} = [U_1,\ldots,U_s]$ are the vectors which define the lower and upper bounds of the search space domain.

Generally, there is no single point $\boldsymbol{x}^*$ which minimizes all k objective functions simultaneously; instead there is a set of solutions which are incomparable and is known as the Pareto set. The basic definitions concerning Pareto optimality are as follows (Simon 2013).

- *Domination*: A point $\boldsymbol{x}^*$ is said to dominate $\boldsymbol{x}$ if the following two conditions hold: (1) $f_i(\boldsymbol{x}^*) \le f_i(\boldsymbol{x})$ for all $i \in [1,k]$, and (2) $f_j(\boldsymbol{x}^*) < f_j(\boldsymbol{x})$ for at least one $j \in [1,k]$.
- *Weak Domination*: A point $\boldsymbol{x}^*$ is said to weakly dominate $\boldsymbol{x}$ if $f_j(\boldsymbol{x}^*) \le f_j(\boldsymbol{x})$ for all $j \in [1,k]$.
- *Nondominated*: A point $\boldsymbol{x}^*$ is said to be nondominated if there is no $\boldsymbol{x}$ that dominates $\boldsymbol{x}^*$.
- *Pareto optimal points*: A Pareto optimal point $\boldsymbol{x}^*$ is a point that is not dominated by any other point in the search space.
- *Pareto set*: Pareto set (P_s) is the set of nondominated points in the search space.
- *Pareto front* (or *Pareto*): Pareto front (P_f) is the set of all function vectors $f(\boldsymbol{x})$ corresponding to the Pareto set.

3.2 Extending single-objective PSO to multi-objective PSO

Several approaches can be found in the literature for applying PSO to MOPs. Some of them such as aggregating approaches and lexicographic ordering methods try to solve a given MOP by defining and solving new SOPs. In aggregating approaches, MOPs are converted to SOPs by combining all objective functions into one objective function. In lexicographic ordering methods, objective functions are ranked based on their importance to the user. Minimization of each objective function is subsequently considered as a new SOP. See (Reyes-Sierra and Coello Coello 2006) and (Coello Coello et al. 2007) for more information.

In this paper, however, a Pareto-based approach is presented in which the single-objective PSO formulation is extended to a multi-objective formulation. Therefore, (1) must be modified so that it can be applied to MOPs because there is no ***pbest*** or ***lbest*** available in a multi-objective approach. The reason is that in MOPs there are Pareto fronts instead of one minimum point. Researchers have defined a variety of methodologies to deal with this problem. See, for example, (Coello Coello and Lechuga 2002); (Reyes-Sierra and Coello Coello 2006); (Sierra and Coello Coello 2005); (Nebro et al. 2009).

Coello Coello and Lechuga (2002) proposed one of the earliest unconstrained continuous multi-objective PSOs called MOPSO. The flight stage of the PSO formulation is based on an inertia weight formulation where inertia weight equal to 0.4 is used. A repository is defined in MOPSO where all nondominated solutions are stored. Leaders are randomly selected from the same repository. OMOPSO (Sierra and Coello Coello 2005) and SMPSO (Nebro et al. 2009) are two highly competitive Pareto-based multi-objective PSO algorithms that can be applied to continuous MOPs. OMOPSO uses one archive of leaders for guiding the swarm. A binary tournament based on the crowding values is used for selecting one leader for each particle. Furthermore, the swarm is divided into three sub-swarms of equal sizes. Uniform and non-uniform mutation operators are applied to the first two sub-swarms, respectively. No mutation operator is applied to the third sub-swarm. OMOPSO utilizes a PSO formulation based on inertia weight

for updating the velocities at each generation. Furthermore, random cognition and social learning factors are used in OMOPSO. If any particle gets out of bounds of the search space, it will be put on the boundaries and the direction of its velocity will be reversed. The same mechanism that was used in OMOPSO for selection of the leaders is also used in SMPSO. However, a constriction factor approach with random social and cognition learning factors are used in SMPSO. If any particle gets out of bounds of the search space, it will be put on the boundaries and its velocity will be reduced by multiplying a factor between 0.001 and 0.1. 15% of particles are mutated by a polynomial mutation operator while the rest of particles are not mutated.

In this paper, a novel approach is proposed to ensure faster convergence while the diversity of the Pareto front is preserved. Most researchers rename the ***lbest*** particle to the leader particle for a multi-objective problem. Selection of the leader and the local best particle has a direct impact on the behavior of the swarm both in the rate of its convergence and in the diversity of the Pareto front. According to the authors' experience, even minor changes in selection of the leader will end up in a totally different behavior. To ensure faster convergence we define the leader in a way that takes advantage of the information available from all positions in the search space that the particles have experienced so far. However, this strategy may result in premature convergence to some local minima. Therefore, other procedures will be incorporated for improving diversity of the archives and enhancing the selection procedure.

In the original PSO, $\boldsymbol{pbest}_i$ represents the best position found so far by the i^{th} particle. For a multi-objective problem, we generalize this concept to P_i set which is defined as the set of nondominated solutions found so far by the i^{th} particle. Subsequently, a particle called $\boldsymbol{psel}_i$ will be selected from P_i for performing the flight stage. Moreover, by assuming that n is the population size and $N \le n-1$ is the number of neighbors randomly selected for each particle at each generation, the concept of $\boldsymbol{lbest}_i$ is also generalized so that it represents the leader of the i^{th} particle ($\boldsymbol{leader}_i$). At each generation and for each individual i, N neighbor P_j sets ($j \in [1, N]$) are selected randomly and a nondominated set called Q_i is extracted from the union of the selected P_j sets. $\boldsymbol{Leader}_i$ will be a particle selected from Q_i by some procedure which will be explained later. Therefore, for a multi-objective problem, the following equations shall be used for the flight stage.

$$\boldsymbol{v}_{\boldsymbol{i}}^{t+1} = \chi\left[\boldsymbol{v}_{\boldsymbol{i}}^{t} + \phi_1 \boldsymbol{r}_1 \circledast (\boldsymbol{psel}_i - \boldsymbol{x}_i) + \phi_2 \boldsymbol{r}_2 \circledast (\boldsymbol{leader}_i - \boldsymbol{x}_i)\right] \tag{3}$$

$$\boldsymbol{x}_{\boldsymbol{i}}^{t+1} = \boldsymbol{x}_{\boldsymbol{i}}^{t} + \boldsymbol{v}_{\boldsymbol{i}}^{t+1} \tag{4}$$

As mentioned before, the constriction factor is meant to prevent velocity blowups. However, the authors have observed better performance for the algorithm proposed in this paper by applying further limits to velocity vectors through (5).

$$v_{i_j}^{t+1} \leftarrow \begin{cases} v_{i_j}^{t+1}, & if \quad \left|v_{i_j}^{t+1}\right| \le v_m^j \\ v_m^j sign\left(v_{i_j}^{t+1}\right), & otherwise \end{cases}, j \in [1, s] \tag{5}$$

where $\boldsymbol{v_m} = \left[v_m^1, \ldots, v_m^s\right]$ is the vector of maximum allowable velocities. PSO systems usually start the search with random or zero valued velocities. Starting the search with zero velocities would indicate that the method should start finding the correct path based on only the cognition and social learning mechanisms. It is not the best choice especially if the population size is small. Because, little data about the search space is provided with small population sizes and the method would have little chance to continue the search with appropriate velocities. It can subsequently result in stagnation in early generations. On the other hand, starting the search with large random velocities is not recommended for solving problems with limited number of function evaluations either. Because, it would need too many generations before the initial random velocities could be corrected by cognition and social learning mechanisms. Therefore, the authors suggest that FC-MOPSO should be started with random velocities not larger than $\boldsymbol{v}_m = 0.1(\boldsymbol{U} - \boldsymbol{L})$. In other words, we prevent the method from starting with large velocities while we also give it some initial velocities for starting the search. The value of 0.1 is selected to represent a small value and to prevent each dimension of each particle from displacing more than 10% of its corresponding search space size at the initial generation. A small value like 0.1 would let the particles escape from stagnation in early generations and correct their paths based on cognition and social learning mechanisms in later generations. For next generations, however, $\boldsymbol{v}_m = rand()(\boldsymbol{U} - \boldsymbol{L})$ is recommended where $rand()$ is a random number between zero and one. A random value, rather than a constant parameter, is used here so that parameter tuning would not be required. (5) does not apply to discrete variable problems (see Section 5). Besides, if a component of an updated particle from (4) gets outside the search space it will be put on the boundary of the search space. In other words, for all $j \in [1, s]$ if it happens that $x_{i_j}^{t+1} < L_j$ or $x_{i_j}^{t+1} > U_j$ then $x_{i_j}^{t+1} = L_j$ or $x_{i_j}^{t+1} = U_j$ will be used, respectively.

In addition to the procedures discussed in the next section for preventing premature convergence, mutation operators will be incorporated for increasing the chance of finding global optimum solutions. Three mutation operators, (1) the polynomial mutation operator (Deb and Deb 2014), (2) the nonuniform mutation operator (Sierra and Coello Coello 2005) and (3) the binary mutation operator presented in Fig. 1 were checked. It was found that the binary mutation operator works the best for our proposed algorithm. In this method, at each

```
For j=1 to s
    If rand( ) ≤ p_m
        For b=1 to B_j
            If rand( ) ≤ ρ_j
                if b^th bit of the binary format for x_ij is equal to zero, convert it to one. Otherwise convert it to zero.
            End if
        End for
    End if
End for
Return x_i.
```

Fig. 1 Pseudo-code of binary mutation operator of the i^{th} particle

generation, after updating the position of each particle, a binary mutation operator is applied to obtain the final position of the particle at that generation. For continuous MOPs, binary representations of each particle's position are used so that the binary mutation operator can be applied. Throughout this paper we accomplish this conversion with a precision of 0.01. That is, the length of strings used for converting continuous variables in the j^{th} dimension of the search space to the binary format is calculated from (6).

$$B_j = floor\left(log\left(\frac{U_j - L_j}{0.01} + 1\right) \Big/ log(2)\right) + 1 \tag{6}$$

Following this procedure, the binary representation used for U_j is a string of length B_j with all its bits equal to one while the binary representation for L_j is a string of length B_j with all its bits equal to zero. Therefore, there will be no chance for a particle to get outside the search space once the mutation is applied. Moreover, the difference between two consecutive binary representations will be equal to $\left(U_j - L_j\right) / \left(2^{B_j} - 1\right)$. Probabilities equal to $p_m = 1/s$ and $\rho_j = 1/B_j$ are used for mutating each element of each particle and each bit of that element, respectively.

3.3 Selection procedure and preservation of diversity

The following aspects are considered for preventing premature convergence and preserving the diversity of the Pareto front. Firstly, the size of P_i archives is restricted. Based on a vast investigation, the number of elements in each P_i set is limited to nine elements in this paper. That is, after updating the position of i^{th} particle at each generation, if the particle is not dominated by any members of P_i set, the particle is added to P_i set. Moreover, if this new member dominates some members in the set, those members will be excluded from P_i. Finally, if the number of members of the set has grown to be more than nine, some members with the least crowding distances will be eliminated to limit the archive size. In a rare case that there is more than one particle in P_i with the least crowding distance, ε-dominance is incorporated to eliminate one of the ten members available in P_i.

A similar procedure is applied to each Q_i set. In other words, after generating Q_i from N selected P_i sets, Q_i will be pruned using crowding distances or ε-dominance concept to limit their size to 100. The concepts of crowding and ε-dominance are defined in Sections 3.3.1 and 3.3.2.

Secondly, the following procedure is suggested for selecting $\boldsymbol{psel}_i$ from P_i and for selecting $\boldsymbol{lsel}_i$ from Q_i. The crowding distances of all members in P_i (or Q_i) are evaluated and the one with the largest crowding distance value is the selected $\boldsymbol{psel}_i$ (or $\boldsymbol{lsel}_i$). If there are more than one member holding the largest crowding distance, a roulette-wheel selection will be employed for preferring one of them. One typical case that leads to equal crowding distances in a set is the situation in which the set consists of only two members. Since the problem is multi-objective, one has to assign a fitness value to each member before applying a roulette-wheel selection. This can be accomplished by the following equation.

$$\bar{f}(\boldsymbol{x}_i) = \left(\sum_{j=1}^{k} w_j \hat{f}_j(\boldsymbol{x}_i)\right) \Big/ \left(\sum_{j=1}^{k} w_j\right) \tag{7}$$

where $\bar{f}(\boldsymbol{x}_i)$ represents the fitness value assigned to particle $\boldsymbol{x}_i$ and w_j refers to the weight factor assigned to the j^{th} objective function. The same w_j's are used in this paper. The hat symbol on objective function values in (7) indicates that normalized objective function values evaluated from (8) are used.

$$\hat{f}_j(\boldsymbol{x}_i) = \frac{f_j(\boldsymbol{x}_i) - \min_{i\in[1,m_i]}\left(f_j(\boldsymbol{x}_i)\right)}{\max_{i\in[1,m_i]}\left(f_j(\boldsymbol{x}_i)\right) - \min_{i\in[1,m_i]}\left(f_j(\boldsymbol{x}_i)\right)}, j\in[1,k] \tag{8}$$

where m_i is the number of particles that are incorporated for normalizing the values of objective functions for the i^{th} particle. For a constrained problem, objective functions will be normalized based on the minimum and maximum values of objective functions in current generation, i.e. $m_i = n$ for $i\in[1, n]$. For unconstrained problems normalization will be restricted to situations that either evaluation of crowding distances or the selection procedure is encountered. Thus, normalization for unconstrained problems will be done only based on the minimum and maximum values available in each P_i (or Q_i) set. In other words, m_i will be the number of particles in each P_i (or Q_i) set for unconstrained problems.

3.3.1 Crowding

Crowding metrics have been used successfully for maintaining diversity of the Pareto front in many EAs. Deb et al. (2002) defined an efficient crowding distance for modifying the fitness values in their NSGA-II algorithm. Their approach does not require parameter tunings unlike other density estimator concepts such as Kernel density estimator (Goldberg and Richardson 1987). This crowding metric is incorporated in the present paper after making some changes to its original definition given by Deb et al. (2002). Moreover, unlike NSGA-II, crowding distances will not be used for modifying the fitness values. The metric is rather utilized in selection procedure and refining P_i and Q_i archives.

In this paper, the crowding distance for each particle, $C(\boldsymbol{x}_i)$, is defined as the Euclidian norm of $d(\boldsymbol{x}_i)$:

$$C(\boldsymbol{x}_i) = |d(\boldsymbol{x}_i)| \tag{9}$$

where

$$d(\boldsymbol{x}_i) = \left[d_1(\boldsymbol{x}_i), \ldots, d_j(\boldsymbol{x}_i), \ldots, d_k(\boldsymbol{x}_i)\right] \tag{10}$$

and

$$d_j(\boldsymbol{x}_i) = 1\Big/2\left[\operatorname{abs}\left(\hat{f}_j(\boldsymbol{x}_i) - \hat{f}_j^{-}(\boldsymbol{x}_i)\right) + \operatorname{abs}\left(\hat{f}_j^{+}(\boldsymbol{x}_i) - \hat{f}_j(\boldsymbol{x}_i)\right)\right], j\varepsilon[1,k] \tag{11}$$

$\hat{f}_j^{+}(\boldsymbol{x}_i)$ and $\hat{f}_j^{-}(\boldsymbol{x}_i)$ are respectively the closest larger value and the closest smaller value to $\hat{f}_j(\boldsymbol{x}_i)$ among m particles. For extreme positions, however, only a larger or a smaller value is available. In such cases $\hat{f}_j^{+}(\boldsymbol{x}_i) = \hat{f}_j^{-}(\boldsymbol{x}_i)$ will be used. Regarding the above definitions, it can be concluded that a particle with a smaller crowding distance lies in a more crowded region of the objective function space and vice versa.

3.3.2 ε-dominance

Another concept that is incorporated for preserving diversity and preventing premature convergence is the ε-dominance idea. In this method, the normalized objective function space is divided into n_e hyperboxes. For a minimization problem, if there is more than one particle in a hyperbox the one closest to the lowest left corner of the hyperbox will be maintained while other particles are eliminated.

Hyperboxes can be formed by dividing each axis of the coordinate system into segments with a size equal to ε_j.

$$\varepsilon_j = \frac{\max\limits_{i\in[1,m_i]}\left(f_j(\boldsymbol{x}_i)\right) - \min\limits_{i\in[1,m_i]}\left(f_j(\boldsymbol{x}_i)\right)}{(n_e)^{1/k}}, j\varepsilon[1,k] \tag{12}$$

If hyperboxes are generated in the normalized objective function space, (12) can be simplified to $\varepsilon_j = 1/(n_e)^{1/k}$ for $j\in[1, k]$. In this paper, n_e equal to 9 and 100 is used for P_i and Q_i sets, respectively. Fig. 2 shows a schematic for hyperboxes in 2D objective function space. Particles shown with filled circles are maintained while the ones represented by unfilled circles are thrown away.

A pseudo-code for unconstrained continuous multi-objective problems is presented in Fig. 3.

4 Constrained continuous FC-MOPSO

In this section, a penalty-based approach is introduced to extend the algorithm proposed in Section 3 to constrained continuous MOPs. In a penalty-based approach, penalized objective functions are frequently minimized instead of the original objective functions. However, a new approach is proposed in this paper which will benefit from the penalized and non-penalized objective functions simultaneously. There are a variety of penalty function definitions available in the literature that can be applied to the method presented here. The authors have found out that the adaptive penalty function suggested by Yen (2009) works very well for the FC-MOPSO algorithm. This adaptive penalty function has the advantage that it takes into account both the amount of constraint violations and the number of constraints violated at each generation. Moreover, with the aid of this penalty function, the information available from both feasible and infeasible regions will be employed to encourage the population to move toward a feasible Pareto front. Let $\varphi(\boldsymbol{x}_i) = [\varphi_1(\boldsymbol{x}_i),\ldots, \varphi_j(\boldsymbol{x}_i),\ldots, \varphi_k(\boldsymbol{x}_i)]$ represent the vector of k penalized objective functions for the i^{th} particle. Then, at each generation t, each component of the penalized objective function is defined as sum of two parts:

$$\varphi_j(\boldsymbol{x}_i) = \Lambda_j(\boldsymbol{x}_i) + \Omega_j(\boldsymbol{x}_i) \tag{13}$$

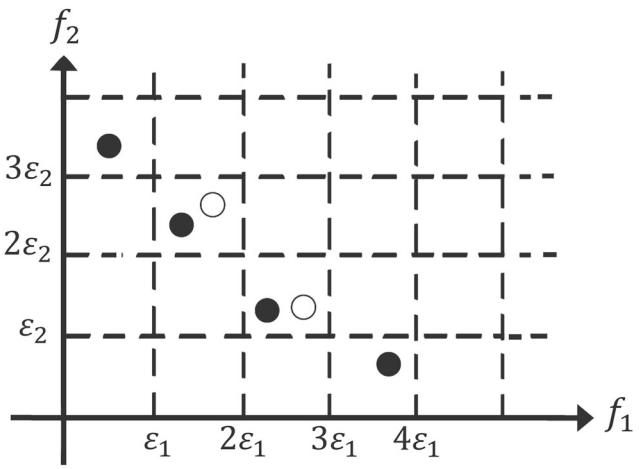


Fig. 2 Schematic of hyperboxes and the concept of ε-dominance in 2D objective function space

Start with n random particles and n random vectors of velocities such that $L_j \le x_{ij} \le U_j$ and $-v_m^j \le v_{ij} \le v_m^j$ for $i \in [1, n]$ and $j \in [1, s]$.
For each particle
    Put $\boldsymbol{x}_i$ into P_i set.
    $\boldsymbol{psel}_i \leftarrow \boldsymbol{x}_i$.
Next i
Evaluate each particle's fitness and create P_f and P_s.
Select N random neighbors for each particle.
For each particle
    Create Q_i using all P_j, $j \in [1, N]$ and limit its size to 100 nondominated individuals by considering crowding and ε-dominance.
    Based on crowding distance or roulette-wheel selection, whichever is applicable, select $\boldsymbol{leader}_i$ from Q_i.
Next i
While termination criterion is not met
    For each particle
        Create a random $\boldsymbol{v}_m$ and update each particle's velocity with Eqs. (3) and (5), respectively.
        Update i^{th} particle position with Eq. (4) and prevent the particle from getting outside the search space.
        Mutate $\boldsymbol{x}_i$.
        Update P_i and limit its size to nine nondominated individuals.
        Based on crowding distance or roulette-wheel selection, whichever is applicable, select $\boldsymbol{psel}_i$ from P_i.
        Update P_f and P_s.
    Next i
    Select N random neighbors for each particle.
    For each particle
        Create Q_i using all P_j, $j \in [1, N]$ and limit its size to 100 nondominated individuals by considering crowding and ε-dominance.
        Based on crowding distance or roulette-wheel selection, whichever is applicable, select $\boldsymbol{leader}_i$ from Q_i.
    Next i
End while
Return P_f and P_s.

Fig. 3 Pseudo-code of the unconstrained continuous FC-MOPSO

where $\Lambda_j(\boldsymbol{x})$ is called the distance value of particle $\boldsymbol{x}$ and is defined in (14).

$$\Lambda_j(\boldsymbol{x_i}) = \begin{cases} \psi(\boldsymbol{x_i}), & if\ r_f = 0 \\ \sqrt{\hat{f}_j(\boldsymbol{x_i})^2 + \psi(\boldsymbol{x_i})^2}, & otherwise \end{cases} \tag{14}$$

where $\psi(\boldsymbol{x})$ represents the average normalized constraint violation and is defined as below.

$$\psi(\boldsymbol{x_i}) = \frac{1}{q} \sum_{j=1}^{q} \frac{c_j(\boldsymbol{x_i})}{c_{max}^j} \tag{15}$$

Start with n random particles and n random vectors of velocities such that $L_j \le x_{ij} \le U_j$ and $-v_m^j \le v_{ij} \le v_m^j$ for $i \in [1, n]$ and $j \in [1, s]$.
For each particle
    Put $\boldsymbol{x}_i$ into P_i set.
    $\boldsymbol{psel}_i \leftarrow \boldsymbol{x}_i$.
Next i
Evaluate the f-values.
Check feasibility of each individual and create P_f and P_s if there are some feasible individuals.
Evaluate $\hat{f}$- and φ-values.
Select N random neighbors for each particle.
For each particle
    Create Q_i based on φ-values corresponding to all P_j, $j \in [1, N]$ and limit its size to 100 nondominated individuals by applying the crowding distance concept or the ε-dominance concept in the space of $\hat{f}$-values corresponding to Q_i.
    In the space of φ-values corresponding to Q_i, apply crowding distance or roulette-wheel selection, whichever is applicable, to select $\boldsymbol{leader}_i$ from Q_i.
Next i
While termination criterion is not met
    For each particle
        Create a random $\boldsymbol{v}_m$ and update each particle's velocity with Eqs. (3) and (5), respectively.
        Update i^{th} particle position with Eq. (4) and prevent the particle from getting outside the search space.
        Mutate x_i.
        Evaluate $f(\boldsymbol{x}_i)$ and check the feasibility of $\boldsymbol{x}_i$.
        If $\boldsymbol{x}_i$ is a feasible point, update P_f and P_s.
        Evaluate $\hat{f}(\boldsymbol{x}_i)$ and $\varphi(\boldsymbol{x}_i)$.
        Update P_i and limit its size to nine by applying crowding or ε-dominance in the space of $\hat{f}$-values corresponding to P_i.
        Based on the crowding distance (or roulette-wheel selection, if applicable) in the space of φ-values corresponding to P_i, select $\boldsymbol{psel}_i$ from P_i.
    Next i
    Select N random neighbors for each particle.
    For each particle
        Create Q_i based on φ-values corresponding to all P_j, $j \in [1, N]$ and limit its size to 100 nondominated individuals by applying crowding distance or ε-dominance in the space of $\hat{f}$-values corresponding to Q_i.
        In the space of φ-values corresponding to Q_i, apply crowding distance or roulette-wheel selection, whichever is applicable, to select $\boldsymbol{leader}_i$ from Q_i.
    Next i
End while
Return P_f and P_s.

Fig. 4 Pseudo-code of the constrained continuous FC-MOPSO

$c_j(\boldsymbol{x}_i)$ is the constraint violation of the j^{th} constraint for the i^{th} particle and c^j_{max} is the maximum constraint violation of the j^{th} constraint in the current population:

$$c_j(\boldsymbol{x_i}) = \begin{cases} max\Big(0, g_j(\boldsymbol{x_i})\Big), & if\ j\in[1,p] \\ max\big(0, |h_j(\boldsymbol{x_i})|-\delta\big), & if\ j\in[p+1,q] \end{cases} \tag{16}$$

$$c^j_{max} = \max_{i\in[1,\ n]} c_j(\boldsymbol{x_i}) \tag{17}$$

where δ is the threshold used for accepting that an equality constraint is satisfied. $\delta = 0.0001$ is usually used in benchmark problems.

The second part of the penalty function is defined as follows.

$$\Omega_j(\boldsymbol{x_i}) = (1-r_f)X_j(\boldsymbol{x_i}) + r_f Y_j(\boldsymbol{x_i}) \tag{18}$$

where $r_f = n_f/n$ is the ratio of feasible solutions in current population and n_f is the number of feasible particles among current population. In the above equation, $X_j(\boldsymbol{x}_i)$ and $Y_j(\boldsymbol{x}_i)$ are two penalty functions as defined in (19) and (20).

$$X_j(\boldsymbol{x_i}) = \begin{cases} 0, & if\ r_f = 0 \\ \psi(\boldsymbol{x_i}), & otherwise \end{cases} \tag{19}$$

$$Y_j(\boldsymbol{x_i}) = \begin{cases} 0, & if\ \boldsymbol{x}_i\ is\ feasible \\ \hat{f}_j(\boldsymbol{x_i}), & otherwise \end{cases} \tag{20}$$

At each generation, after evaluating fitness values (f-values) for the population, normalized fitness values ($\hat{f}$-values) are obtained from (8). Finally, having checked the feasibility of each particle in the population, penalized objective function values (φ-values) can be computed for each individual in the population. Fig. 4 outlines the present FC-MOPSO algorithm for handling continuous constrained MOPs. In this figure, P_f represents the Pareto front obtained by constructing a nondominated set from feasible individuals in the space of f-values. The Pareto set corresponding to P_f is called P_s.

5 A discrete version of FC-MOPSO

There are very limited studies that tried to address discrete multi-objective PSO algorithms. One of the most recent of such studies was accomplished by Tong et al. (2016) where an extension of MDPSO (Chowdhury et al. 2013) was recommended for solving problems with mixed discrete design variables. The flight stage in MDPSO is performed using a continuous PSO formulation. The Nearest Vertex Approach (NVA) is subsequently used for approximating the discrete-domain position of a particle. Another example of such studies is the discrete PSO developed by Chen et al. (2009) that utilizes an inertia-based formulation of PSO. The method incorporates crossover and mutation parameters in genetic algorithm so that the flight stage formulation of PSO can be applied to discrete problems.

Here, the algorithm proposed in previous sections is modified so that constrained/unconstrained MOPs with discrete or mixed discrete design variables can also be solved. In fact, only minor modifications is required before FC-MOPSO can be applied to such problems. A discrete or mixed discrete MOP is the case that occurs in design of most real-world civil structures. Steel buildings, trusses and some types of fixed offshore platforms are examples where hot rolled steel sections are used. Therefore, sizing variables in steel structures are discrete in practice. Sizing variables in concrete structures, however, deal with mixed design variables. For example, dimensions of beam and column sections can be represented by discrete values that are practical and easy for construction purposes while the ratio of longitudinal steel reinforcements can be regarded as continuous design variables. In a more practical approach, like the one performed by Paya et al. (2008), these two types of design variables, namely the ratio of longitudinal reinforcements and cross sectional dimensions, can be merged together to constitute a discrete design variable. Consequently, a set of predefined sections with predefined dimensions and steel ratios, which meet the requirements of building design codes, will be available for the optimization process. A similar approach can be adopted for other design variables such as lateral reinforcements. The above discussion reveals that adopting discrete design variables in civil structures has two main advantages: (1) a discrete problem definition is more realistic (2) the search space becomes smaller which is easier to search while impractical solutions are already eliminated from the set of solutions.

Soon after introducing the continuous version of PSO, Kennedy and Eberhart (1997) proposed a discrete binary version of the continuous counterpart which will be employed in

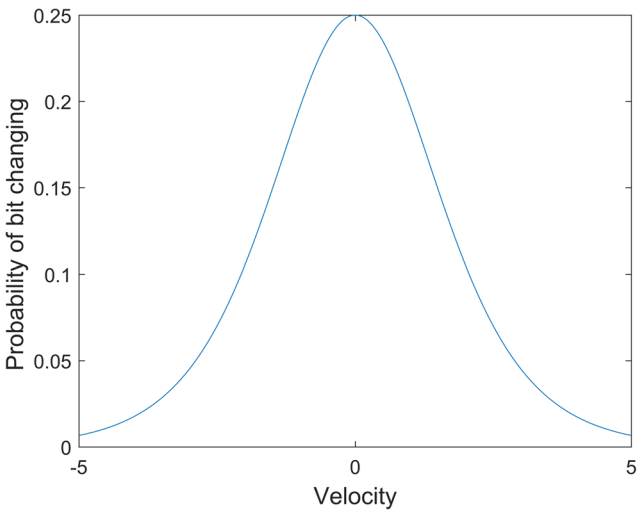


Fig. 5 Probability of bit changing selected for the discrete FC-MOPSO

Table 1 Results for benchmark MOPs after 30 runs

Problem	Number of design variables	Number of objectives	Number of constraints		OMOPSO		SPEA2		NSGA-II		SMPSO		FC-MOPSO	
					I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$
ZDT1	30	2	0	Mean	0.739	0.480	0.610	0.511	0.856	0.531	0.554	0.506	**0.953**	**0.344**
				Std.	0.140	0.065	0.121	0.070	0.150	0.074	0.164	0.058	0.088	0.054
ZDT2	30	2	0	Mean	0.739	0.898	0.473	0.796	0.582	0.949	0.527	0.849	**0.818**	**0.503**
				Std.	0.268	0.110	0.202	0.088	0.222	0.039	0.290	0.141	0.177	0.357
ZDT3	30	2	0	Mean	0.843	0.552	0.711	0.499	0.851	0.461	0.693	0.543	**0.962**	**0.403**
				Std.	0.126	0.077	0.144	0.068	0.107	0.061	0.136	0.093	0.067	0.057
ZDT4	10	2	0	Mean	0.459	0.667	0.253	0.538	**0.678**	0.739	0.527	**0.356**	0.358	0.838
				Std.	0.177	0.161	0.118	0.118	0.108	0.180	0.189	0.224	0.148	0.164
ZDT6	10	2	0	Mean	0.608	0.584	0.456	0.866	0.668	0.952	0.294	0.786	**0.832**	**0.381**
				Std.	0.302	0.207	0.228	0.061	0.307	0.037	0.154	0.084	0.219	0.263
TNK	2	2	2	Mean	0.918	**0.130**	0.911	0.332	**0.959**	0.338	0.863	0.191	0.849	0.198
				Std.	0.051	0.058	0.104	0.161	0.046	0.331	0.101	0.090	0.083	0.117
Golinski	7	2	11	Mean	0.793	**0.051**	0.578	0.430	0.771	0.249	0.718	0.152	**0.871**	0.087
				Std.	0.107	0.026	0.264	0.253	0.206	0.174	0.224	0.152	0.147	0.060
CONSTR	2	2	2	Mean	0.812	0.076	**0.965**	0.153	0.819	0.155	0.782	**0.065**	0.879	0.072
				Std.	0.101	0.032	0.093	0.070	0.141	0.078	0.102	0.014	0.077	0.016
Srinivas	2	2	2	Mean	**0.980**	**0.041**	0.941	0.083	0.761	0.133	0.956	0.052	0.947	0.059
				Std.	0.021	0.016	0.061	0.056	0.064	0.026	0.035	0.019	0.036	0.016
Water	3	5	7	Mean	0.863	0.176	**0.907**	0.218	0.278	0.603	0.716	0.252	0.536	**0.174**
				Std.	0.126	0.031	0.133	0.063	0.152	0.113	0.176	0.050	0.181	0.036
CTP2	2	2	1	Mean	0.854	**0.087**	0.868	0.182	0.860	0.169	**0.892**	0.125	0.868	0.115
				Std.	0.078	0.018	0.129	0.078	0.115	0.043	0.088	0.050	0.098	0.035
CTP3	2	2	1	Mean	0.869	**0.068**	0.892	0.170	0.852	0.150	**0.902**	0.103	0.804	0.149
				Std.	0.069	0.016	0.098	0.079	0.104	0.061	0.074	0.050	0.105	0.049
CTP4	2	2	1	Mean	0.702	0.532	0.536	**0.405**	0.543	0.429	**0.791**	0.773	0.644	**0.405**
				Std.	0.221	0.221	0.201	0.083	0.188	0.108	0.220	0.192	0.212	0.166
CTP5	2	2	1	Mean	**0.709**	0.673	0.559	0.425	0.651	0.429	0.697	0.693	0.625	**0.258**
				Std.	0.209	0.216	0.215	0.177	0.246	0.155	0.196	0.177	0.163	0.057
CTP6	2	2	1	Mean	0.592	**0.167**	0.770	0.246	**0.840**	0.244	0.462	0.177	0.744	0.198
				Std.	0.163	0.087	0.230	0.164	0.165	0.154	0.177	0.056	0.199	0.109

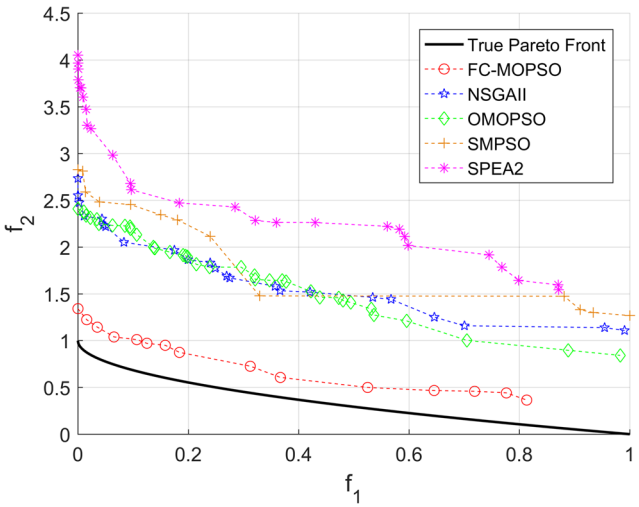


Fig. 6 Pareto fronts for ZDT1 problem

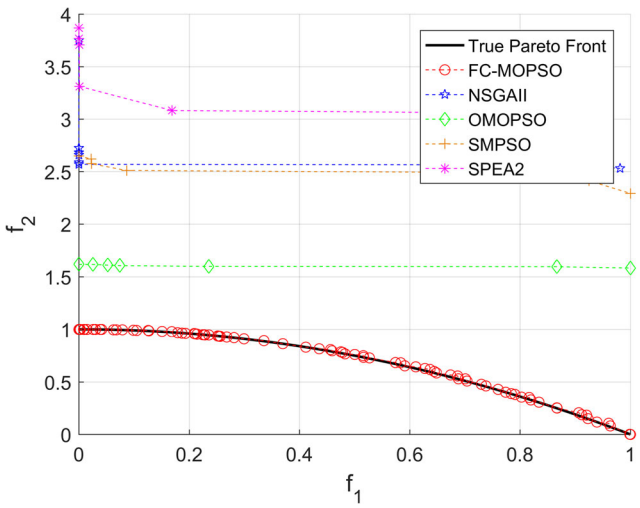


Fig. 7 Pareto fronts for ZDT2 problem

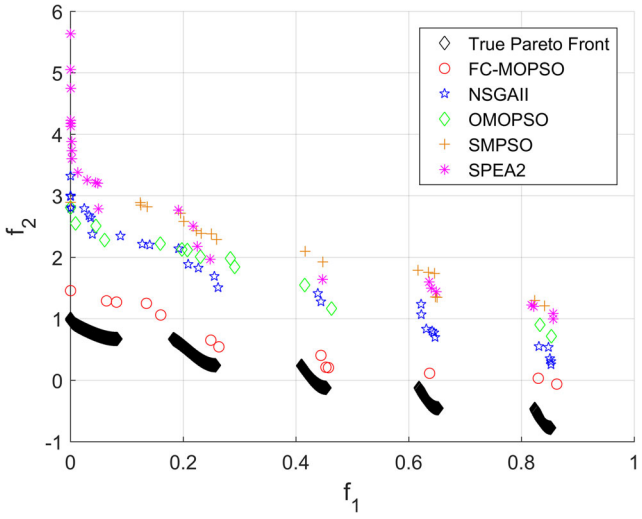


Fig. 8 Pareto fronts for ZDT3

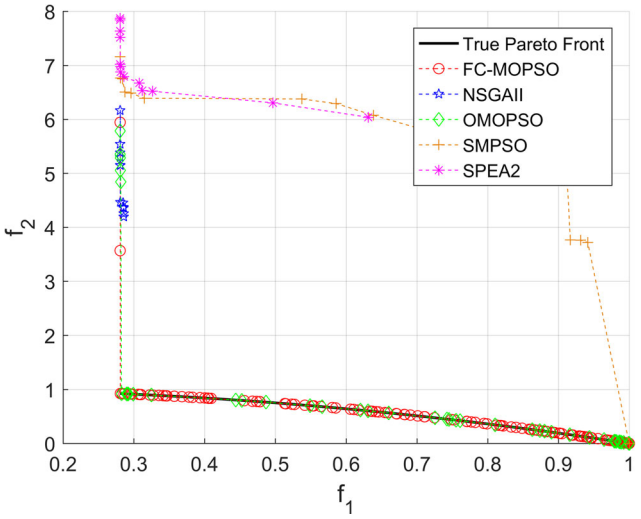


Fig. 10 Pareto fronts for ZDT6 problem

this paper. Let $\boldsymbol{D}_j = \left[D_j^1, \ldots, D_j^{\mu_j}\right], j \in [1, s]$ be the vector including the μ_j discrete values available for the j^{th} design variable and μ_j refer to the number of discrete variables in the j^{th} dimension of the search space. Therefore, the number of bits required for a binary representation of design variables in the j^{th} dimension will be equal to $B_j = \text{floor}(\log(\mu_j)/\log(2)) + 1$. This way, D_j^1 will be defined by a string of length B_j where all its bits are zeroes and D_j^2 is defined with a string that its first bit from the right side is equal to one while other bits are zeroes and so on. It should be noted that $D_j^{\mu_j}$ is not defined by a string with all its bits equal to one. Hence, particles shall be put on the boundaries if they go outside the search space during mutation. Moreover, velocities will be defined in terms of changes of probabilities. In other words, if $x_{i_j,b}$ ($j \in [1, s]$ and $b \in [1, B_j]$) refers to the b^{th} bit of the j^{th} component of the i^{th} particle, then $v_{i_j,b}$ is the corresponding bit of the velocity and represents the probability that $x_{i_j,b}$ takes a value equal to one.

However, $v_{i_j,b}$ can also take values outside the interval [0, 1]. Therefore, to accomplish the interpretation in which velocities are treated as probabilities, an appropriate mapping such as sigmoidal function should be employed so that the probability remains in the interval [0, 1]. The variant of sigmoidal function that returns values restricted to the interval [0, 1] is defined as $S(x) = 1/(1 + \exp(-x))$. Therefore, the FC-MOPSO algorithm defined in Figs. 3 and 4 is still applicable with two modifications. Firstly, at each generation (3) and (4) shall be replaced with (21) and (22), respectively. That is, for each bit $b \in [1, B_j]$ of the j^{th} component of the i^{th} particle we have:

$$v_{i_j,b}^{t+1} = \chi\left[v_{i_j,b}^t + \phi_1 r_1\left(psel_{i_j,b} - x_{i_j,b}^t\right) + \phi_2 r_2\left(leader_{i_j,b} - x_{i_j,b}^t\right)\right] \tag{21}$$

$$x_{i_j,b}^{t+1} = \begin{cases} 1, & if\ rand(\,) < S\left(v_{i_j,b}^{t+1}\right) \\ 0, & otherwise \end{cases} \tag{22}$$

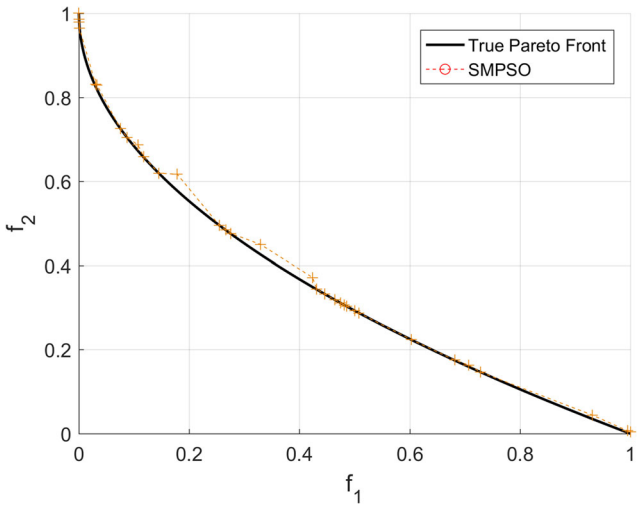


Fig. 9 Pareto fronts for ZDT4

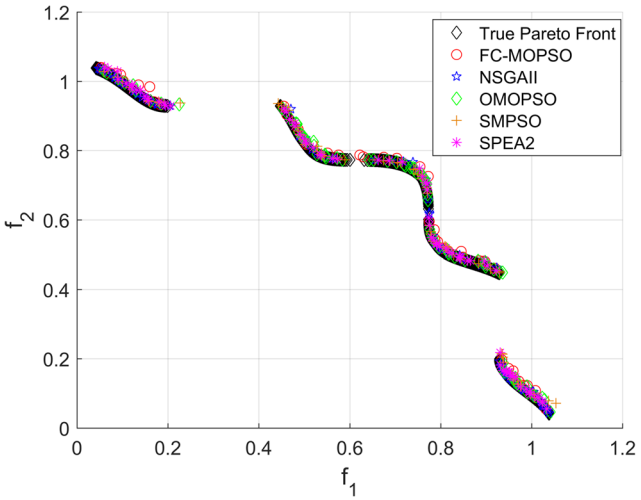


Fig. 11 Pareto fronts for TNK problem

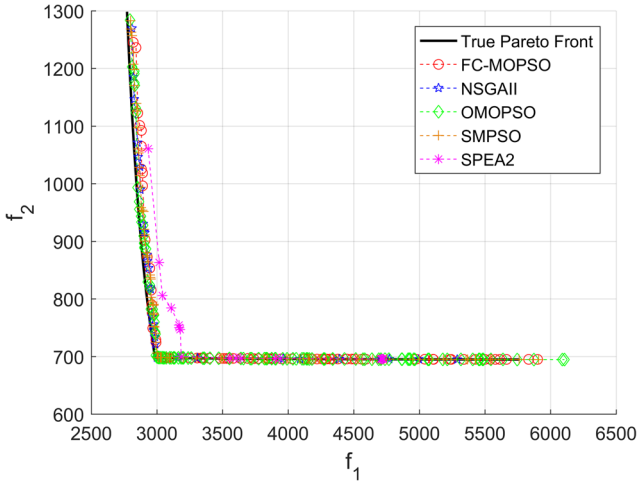


Fig. 12 Pareto fronts for Golinski problem

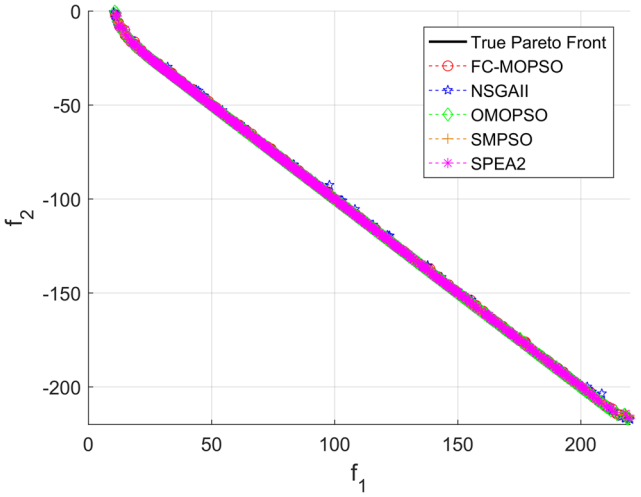


Fig. 14 Pareto fronts for Srinivas problem

Secondly, limitations on maximum velocity will not be applied except for starting the algorithm. That is, random velocity vectors with their bits limited to the interval [−4, 4] are used for initialization of the algorithm while no v_m will be applied to limit the velocities afterwards.

The probability that a bit will be changed is equal to $p\left(x_{i_j,b}\right) = S\left(v_{i_j,b}\right)\left[1-S\left(v_{i_j,b}\right)\right]$ and is depicted in Fig. 5.

6 Test problems and discussion of results

6.1 Benchmark MOPs

15 well-known benchmark problems are employed for validating the performance of continuous FC-MOPSO. These problems include five unconstrained and ten constrained benchmark MOPs. The unconstrained MOPs consist of ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6. These problems have been designed by Zitzler et al. (2000) to assess performance of optimization algorithms in dealing with different types of difficulties in finding the true Pareto front. ZDT1 has a convex Pareto front while ZDT2 is the nonconvex counterpart of ZDT1. ZDT3 represents a problem with discreteness feature because its Pareto front consists of several noncontiguous convex parts. ZDT4 introduces a problem that contains 21^9 local Pareto front. Therefore, it is suitable for validating the ability of an algorithm to deal with multimodality. As noted by Zitzler et al. (2000), ZDT6 is a test problem with nonuniformity in the search space. The Pareto fronts of this problem are nonuniformly distributed along the global Pareto front. Furthermore, the density of solutions is minimum near the Pareto front and is maximum away from it. The constrained MOPs considered in this section consist of the TNK problem (Tanaka et al. 1995), the speed reducer design problem (Kurpati et al. 2002) which is a multi-objective

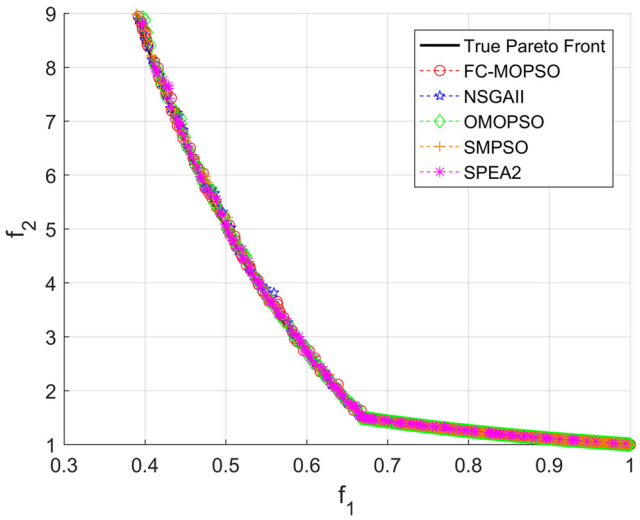


Fig. 13 Pareto fronts for CONSTR problem

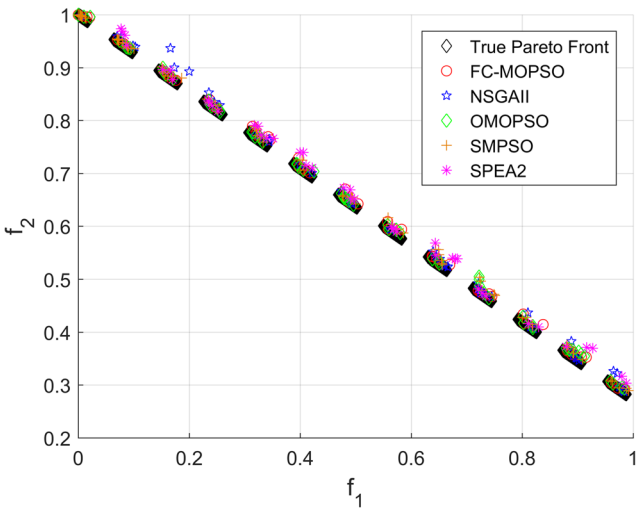


Fig. 15 Pareto fronts for CTP2 problem

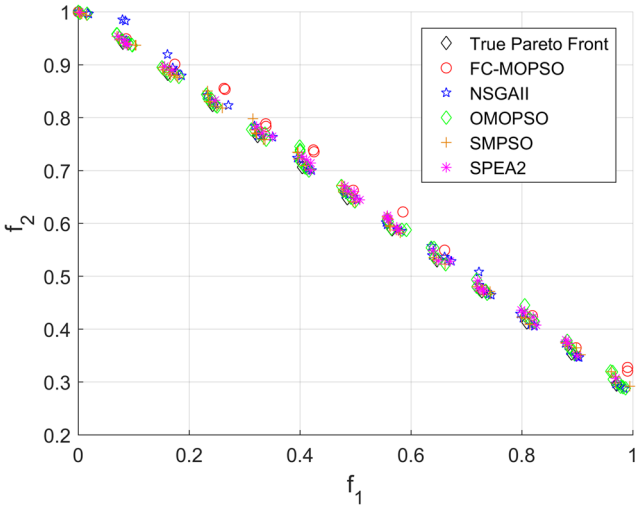


Fig. 16 Pareto fronts for CTP3 problem

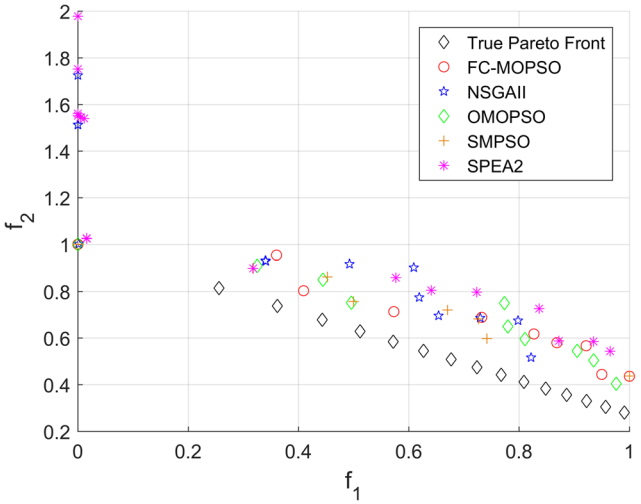


Fig. 18 Pareto fronts for CTP5 problem

variant of the SOP originally defined by Golinski (1970), CONSTR (Deb et al. 2002), Srinivas problem (problem *F*3 in (Srinivas and Deb 1994)), water resource planning problem defined by Ray et al. (2001) and CTP2, CTP3, CTP4, CTP5 and CTP6 (Deb et al. 2001). The CTP test suite consists of problems that cause difficulties either in the entire search space or near the Pareto front.

The 15 benchmark problems are solved by FC-MOPSO and are verified against four well-known algorithms available in jMetal (Durillo and Nebro 2011). There are a variety of algorithms available in jMetal. However, the authors found out that four of them yield the best results within the limited number of function evaluations focused in this paper. Therefore, OMOPSO (Sierra and Coello Coello 2005), SMPSO (Nebro et al. 2009), NSGA-II (Deb et al. 2002) and SPEA2 (Zitzler et al. 2002) are to be employed.

Since EAs are random-based, the results are commonly presented after performing some Monte Carlo simulations. Hence, the results presented here are those obtained after 30 runs ($M = 30$) with the same starting random numbers for all optimizers. For all fifteen benchmark problems, the algorithms were run with a population size of 10 ($n = 10$) and the termination criteria was set to be 400 function evaluations or 40 generations equivalently ($g = 40$). Other parameters for the algorithms in jMetal such as mutation and crossover probability are reasonably left to be their default values in jMetal. Throughout this paper our FC-MOPSO algorithm uses a neighborhood size equal to $N = 9$ for cases with $n = 10$ and $N = 4$ for cases with $n = 30$.

There are many metrics available in the literature that can be employed for comparing two Pareto fronts but it should be noted that none of them is always reliable (Yan et al. 2007). Therefore, more than one metric is commonly used for comparing the performance of algorithms. Here, the hypervolume indicator (I_H) is used for estimating the diversity of a Pareto and its closeness to the true Pareto. The

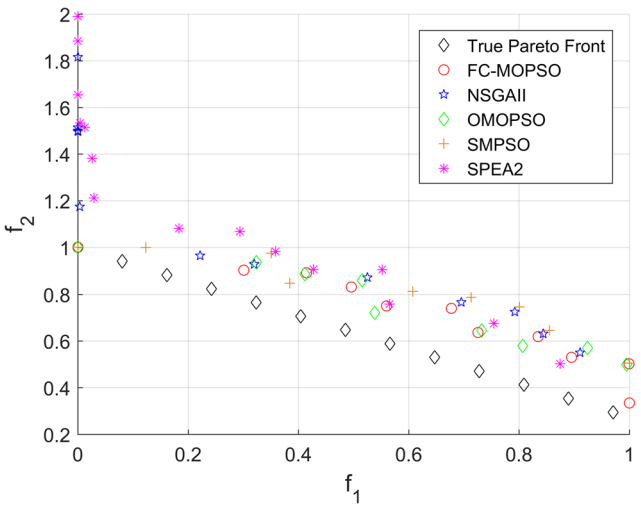


Fig. 17 Pareto fronts for CTP4 problem

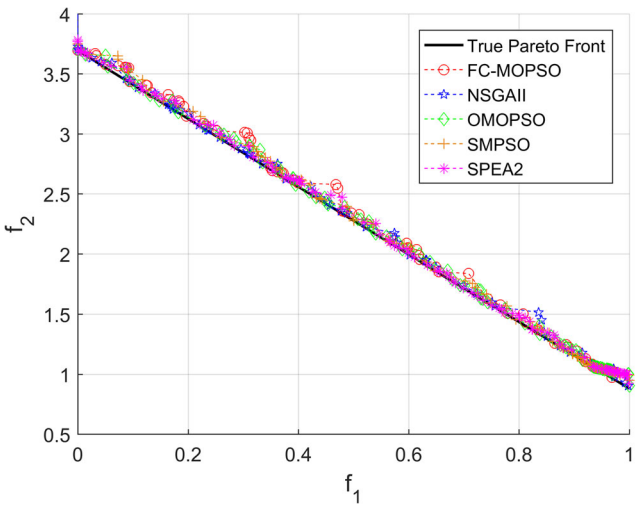


Fig. 19 Pareto fronts for CTP6 problem

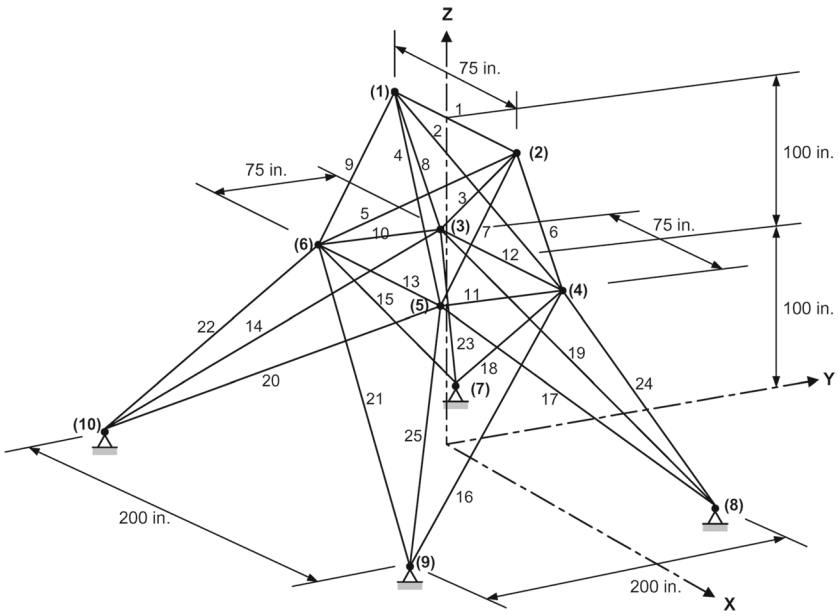


Fig. 20 Schematic of the spatial 25-bar truss structure

additive unary epsilon indicator ($I_{\varepsilon+}$) is also utilized as another indicator of closeness to the true Pareto. A definition for hypervolume of a Pareto front is given in (23) where n_p is the number of points in the Pareto front and $\boldsymbol{r} = [r_1, r_2, \ldots, r_k]$ is a reference point such that it is dominated by all points in the Pareto front. Other parameters in (23) have already been defined in previous sections.

$$I_H(P_f) = \frac{1}{n_p} \sum_{j=1}^{n_p} \prod_{i=1}^{k} \left(r_i - f_i(\boldsymbol{x}_j)\right) \tag{23}$$

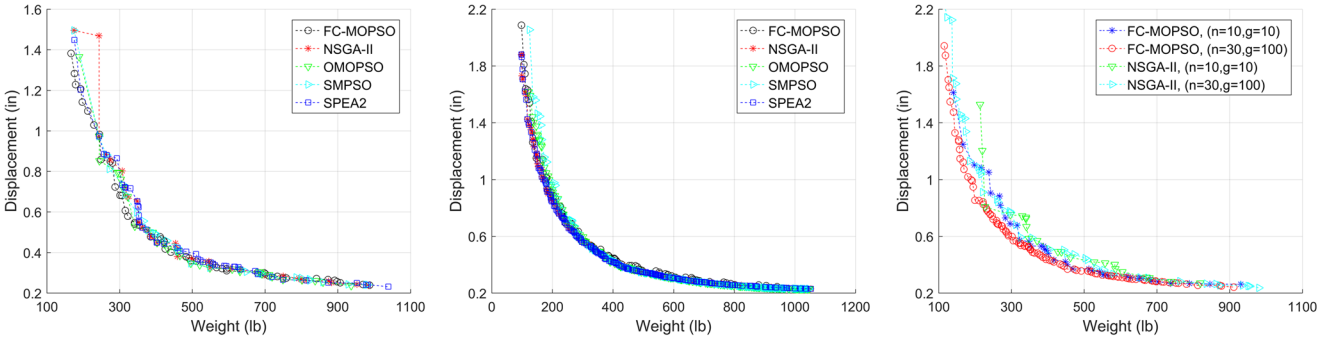
In this paper, for each problem, all Pareto fronts from all M simulations besides the true Pareto of the problem are put together. This new set is normalized for each of its objective functions by using (8). These normalized Pareto fronts are utilized for evaluation through (23) with a reference point with all its elements set equal to 1.01. At each simulation, the Pareto fronts are normalized based on the maximum hypervolume in that simulation and the results are reported. Generally speaking, a higher value of I_H indicates a more preferred Pareto regarding its diversity and closeness to the true Pareto. For the case of $I_{\varepsilon+}$ indicator, on the other hand, the normalized Pareto fronts are employed to evaluate the metric for each Pareto in comparison to the true Pareto. The reader may refer to (Fonseca et al. 2005) for a definition of $I_{\varepsilon+}$. The smaller is the value of $I_{\varepsilon+}$, the closer it is to the true Pareto. With the above mentioned procedure, I_H and $I_{\varepsilon+}$ can take values between zero and one. Table 1 represents the mean and standard deviations for the problems discussed. Bolded values in Table 1 show the best result obtained among all five algorithms for each problem. Figs. 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, and 19 illustrate the best Pareto fronts achieved after 30 runs for all problems except for the Water problem that has five objective functions and cannot be plotted. From Table 1, it can be verified that FC-MOPSO has the highest number of best results among all algorithms. Moreover, Figs. 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, and 19 along with Table 1 show that FC-MOPSO absolutely outperforms other optimizers in ZDT test suite problems except for ZDT4 that deals with multimodality. ZDT problems are of unconstrained type. Therefore, the excellent performance of FC-MOPSO in ZDT problems indicates that extending the original single-objective PSO to FC- MOPSO in Section 3 has been quite successful. Note that none of the optimizers considered in this paper except for SMPSO could converge to the true Pareto front of ZDT4 within 400 function evaluations. The results of these optimizers have been so far away from the true Pareto front that it was not possible to illustrate them in Fig. 9. Fortunately, structural engineering optimization problems do not include high multimodality.

Table 2 Load conditions for 25-bar truss

Node	P_x (*kips*)	P_y (*kips*)	P_z(*kips*)
1	1.0	−10	−10
2	0.0	−10	−10
3	0.5	0.0	0.0
6	0.6	0.0	0.0

6.2 Truss design examples

Five truss design examples are solved to analyze the performance of FC-MOPSO in structural optimization problems. Both discrete and continuous variants of FC- MOPSO are adopted for solving these problems. Furthermore, a problem with mixed discrete and continuous design variables is solved



(a) Continuous solutions, n=10, g=10 (b) Continuous solutions, n=30, g=100 (c) Discrete solutions

Fig. 21 Comparison of the Pareto fronts obtained by different algorithms in the 25-bar truss problem

in Section 6.2.5. All continuous problems are also solved by the following algorithms: OMOPSO, SMPSO, NSGA-II and SPEA2. Discrete variants of the problems are solved by NSGA-II. Pareto fronts obtained from FC-MOPSO and other optimizers are reported after 20 runs for problems with two objective functions. For problems with more than two objective functions, the results are presented after 10 runs. All problems are solved for two cases: (1) $n = 10$, $g = 10$ and (2) $n = 30$, $g = 100$. It should be reminded that the discrete search space is a subdomain of the continuous search space. Hence, potentially, the solutions of a continuous optimization problem can outperform discrete solutions since some parts of the true Pareto front might correspond to regions of the search space that are not available in discrete search space.

6.2.1 Spatial 25-bar truss

The spatial 25-bar truss shown in Fig. 20 was studied as an SOP in many researches such as (Lee and Geem 2004) and (Talatahari et al. 2012). In this paper, a discrete multi-objective approach to this problem that was defined by Luh and Chueh (2004) is considered. It is desired to minimize two objective functions simultaneously: (1) the weight of the structure and (2) the displacement of node 1 in Y-direction. The constraints are defined such that the principal stress in no element exceeds an allowable stress equal to $\sigma_a = \pm 40$ *ksi*. Furthermore, available discrete cross sectional areas are assumed to be D = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4] (in^2). The problem is also solved by considering continuous cross-sectional areas larger than 0.01 in^2 and less than 3.4 in^2. There are eight sizing variables deriving from the following member grouping: (1) A_1, (2) $A_2 \sim A_5$, (3) $A_6 \sim A_9$, (4) $A_{10} \sim A_{11}$, (5) $A_{12} \sim A_{13}$, (6) $A_{14} \sim A_{17}$, (7) $A_{18} \sim A_{21}$, and (8) $A_{22} \sim A_{25}$. A modulus of elasticity equal to $E = 10$ *Msi* and a density equal to $\rho = 0.1$ *lb./in*^{3} is used for all members. Loading conditions for this truss are summarized in Table 2.

Fig. 21 compares the results obtained by FC-MOPSO with those from the other optimizers. I_H and $I_{\varepsilon+}$ indicators for 3000 and 100 function evaluations are reported in Tables 3 and 4, respectively. Similarly, comparisons between discrete solutions found by FC-MOPSO with those from NSGA-II are reported in Table 5.

For the case of 3000 function evaluations, Table 3 shows that the continuous FC-MOPSO outperforms all other optimizers as far as it concerns I_H indicator. From this table, it can also be confirmed that FC-MOPSO is the second best optimizer regarding the $I_{\varepsilon+}$ metric. The difference between $I_{\varepsilon+}$ for FC-MOPSO and the best $I_{\varepsilon+}$, which was found by SPEA2, is very small. Table 4 and Fig. 21a show that for very small number of function evaluations (namely, one hundred evaluations) FC-MOPSO

Table 3 Performance comparison of continuous solutions from FC-MOPSO with other algorithms for truss examples (n = 30, g = 100)

Problem		OMOPSO		SPEA2		NSGA-II		SMPSO		FC-MOPSO	
		I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$
25-bar truss	Mean	0.822	0.076	0.908	**0.041**	0.877	0.046	0.775	0.084	**0.975**	0.045
	Std.	0.064	0.024	0.053	0.026	0.061	0.016	0.069	0.025	0.036	0.009
72-bar truss	Mean	0.698	0.115	0.807	0.148	0.656	0.185	0.691	**0.111**	**0.860**	0.144
	Std.	0.049	0.013	0.037	0.054	0.058	0.017	0.037	0.012	0.044	0.023
120-bar truss	Mean	0.871	0.113	**0.970**	0.051	0.928	**0.050**	0.867	0.121	0.950	0.086
	Std.	0.071	0.040	0.039	0.022	0.045	0.020	0.074	0.041	0.041	0.015

Table 4 Performance comparison of continuous solutions from FC-MOPSO with other algorithms for truss examples (n = 10, g = 10)

Problem		OMOPSO		SPEA2		NSGA-II		SMPSO		FC-MOPSO	
		I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$
25-bar truss	Mean	0.813	0.227	0.911	0.264	0.873	0.237	0.810	0.219	**0.929**	**0.181**
	Std.	0.114	0.061	0.064	0.066	0.091	0.068	0.101	0.051	0.085	0.066
72-bar truss	Mean	0.944	**0.232**	**0.987**	0.314	0.845	0.310	0.923	0.250	0.895	0.261
	Std.	0.029	0.051	0.025	0.047	0.104	0.086	0.047	0.056	0.070	0.047
120-bar truss	Mean	0.784	0.273	0.830	0.261	0.889	0.236	0.811	0.274	**0.892**	**0.197**
	Std.	0.137	0.063	0.092	0.059	0.101	0.077	0.093	0.062	0.059	0.040

is the best optimizer regarding both I_H and $I_{\varepsilon+}$ indicators. In other words, the relative performance of FC-MOPSO, compared with other optimizers studied, increases when a smaller number of function evaluations is used.

As it can be seen from Table 5 and Fig. 21c, for both cases of 100 and 3000 function evaluations, discrete solutions found by FC-MOPSO outperform those from NSGA-II with a remarkable difference. It is noted that the value of I_H relating to NSGA-II for 100 evaluations is larger than I_H for 3000 evaluations which may seem counterintuitive. The reason is that, as it can be seen from Fig. 21c, most points found by NSGA-II in 100 function evaluations are gathered in a small central portion of the true Pareto front. Such situations can result in delusively high values of I_H. That is why, as it was already discussed in Section 6.1, the results should always be justified with more than one metric along with visual inspections whenever it is possible. For this case, the $I_{\varepsilon+}$ indicator does not yield counterintuitive results since its value for 100 function evaluations is smaller than its value for 3000 function evaluations.

The small standard deviation values of FC-MOSO reported in Tables 3, 4, and 5 confirm that the algorithm have a consistent behavior in 20 runs.

6.2.2 Spatial 56-bar truss

The spatial 56-bar truss shown in Fig. 22 is considered as an example of structural MOPs with more than two objective functions. As a discrete optimization problem, practical equal leg double angles as listed in Table 6 are used. Accordingly, cross sectional areas between 200 mm^2 and 2000 mm^2 should be used for a continuous problem. Loading conditions for this truss consist of 4 kN in the Y-direction and 30 kN in the Z-direction at node 1 while all other nodes are loaded with 4 kN in the Y-direction and 10 kN in the Z-direction. The objective functions are to minimize the weight of the structure, the total displacement of node 1 and the first natural frequency of the truss. Note that these objective functions are in conflict with one another. The mass matrix of the truss is assembled by adopting lumped mass formulation for truss elements. The vertical displacements of nodes 4, 5, 6, 12, 13 and 14 is limited to ±40 mm while the displacement in the Y-direction for node 8 is not allowed to exceed ±20 mm. The modulus of elasticity and material density are assumed to be E = 210 GPa and ρ = 7800 kg/m^3, respectively.

Fig. 23 illustrates the results obtained from FC-MOPSO along with the optimum solutions obtained by other optimizers. Fig. 23a and b show the best continuous Pareto fronts obtained by different optimizers for 100 and 3000 function evaluations, respectively. It can be observed that FC-MOPSO could find an acceptable and diverse approximation of the Pareto front while other optimizers have failed to converge to such solutions.

Fig. 23b shows that the continuous FC-MOPSO almost approximated the whole Pareto front while the results of other optimizers cover only a small portion of the Pareto front.

Table 5 Performance comparison of discrete solutions from FC-MOPSO with NSGA-II for truss examples

Problem		NSGA-II (n = 10, g = 10)		FC-MOPSO (n = 10, g = 10)		NSGA-II (n = 30, g = 100)		FC-MOPSO (n = 30, g = 100)	
		I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$	I_H	$I_{\varepsilon+}$
25-bar truss	Mean	0.810	0.247	**0.918**	**0.154**	0.672	0.142	**0.998**	**0.035**
	Std.	0.102	0.055	0.068	0.041	0.100	0.044	0.007	0.008
72-bar truss	Mean	0.943	0.242	**0.988**	**0.184**	0.713	0.201	**0.930**	**0.087**
	Std.	0.054	0.023	0.018	0.035	0.061	0.033	0.041	0.009
120-bar truss	Mean	**0.779**	0.464	0.769	**0.242**	**0.967**	0.186	0.928	**0.065**
	Std.	0.193	0.113	0.118	0.062	0.055	0.040	0.063	0.017

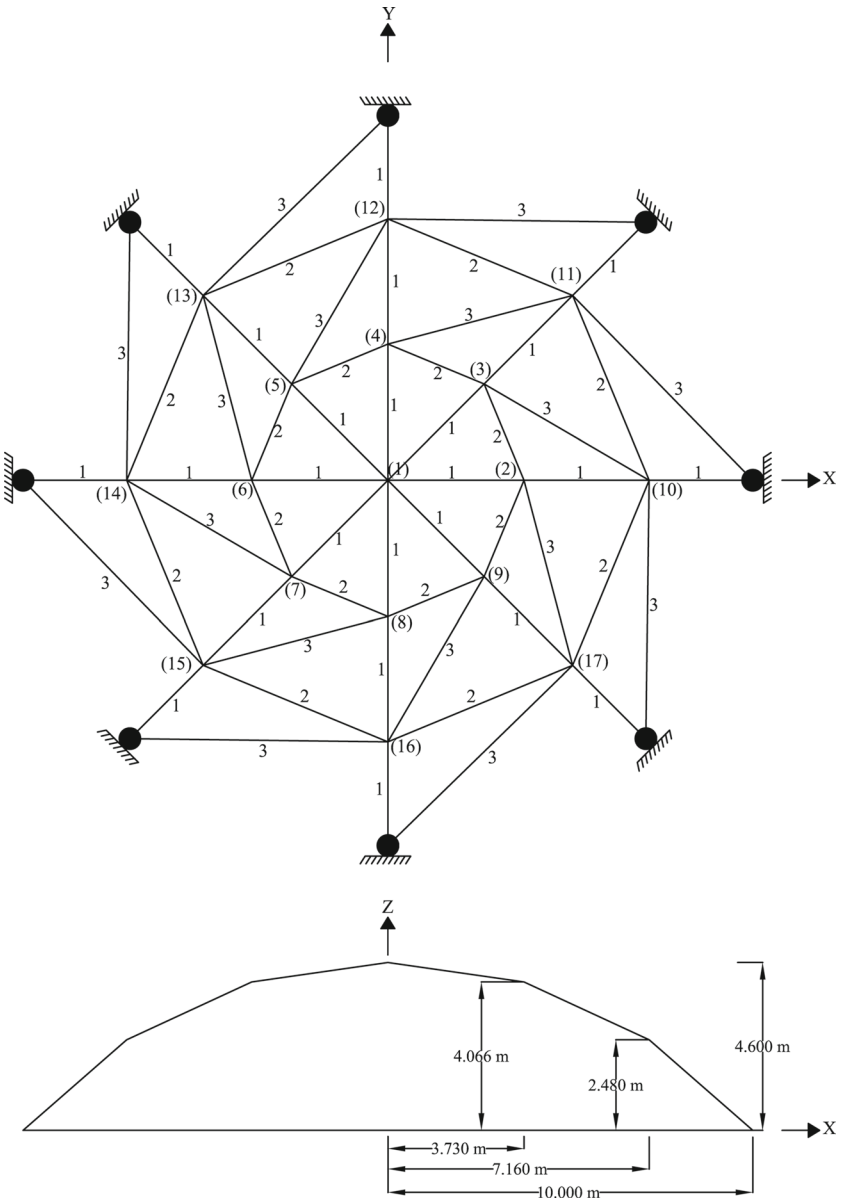


Fig. 22 Schematic of the spatial 56-bar truss structure

Similarly, Fig. 23a shows that for 100 function evaluations and in comparison to other optimizers, FC-MOPSO has managed to find better approximation of the Pareto front.

Results from discrete solutions are illustrated in Fig. 23c. This figure shows that FC-MOPSO could yield much better solutions than NSGA-II. Especially for 100 function evaluations, NSGA-II approximates only a very small portion of the Pareto front in comparison to FC-MOPSO where the whole Pareto front is well approximated.

FC-MOPSO preserves the best experiences of each particle in P_i sets. This preservation along with the techniques that are carried out for pruning the archives and proper selection of the leaders gives FC-MOPSO a better chance for exploring the search space in comparison to other optimizers. Note that using the I_H and $I_{\varepsilon+}$ indicators for comparing two Pareto fronts can be counter-intuitive if there is a great difference in the number of the Pareto optimal points. The number of Pareto optimal points found by FC-MOPSO at each run is much larger than those from other optimizers for this problem as well as the optimization problem defined in Section 6.2.5. Hence, the I_H and $I_{\varepsilon+}$ indicators are not evaluated for these problems.

6.2.3 Spatial 72-bar truss

Another well-known optimization test problem often considered in the literature is the weight optimization of the spatial 4-story 72-bar truss shown in Fig. 24. See, for example, (Adeli and Kamal 1986); (Erbatur et al. 2000); (Lee and Geem 2004); (Jansen and Perez 2011). In this study, discrete sizing optimization is carried out by using the first 31 sections defined in Table 6. For a continuous solution, cross sectional areas between 0.1 in^2 and 2.5 in^2 are utilized. Four objective functions are

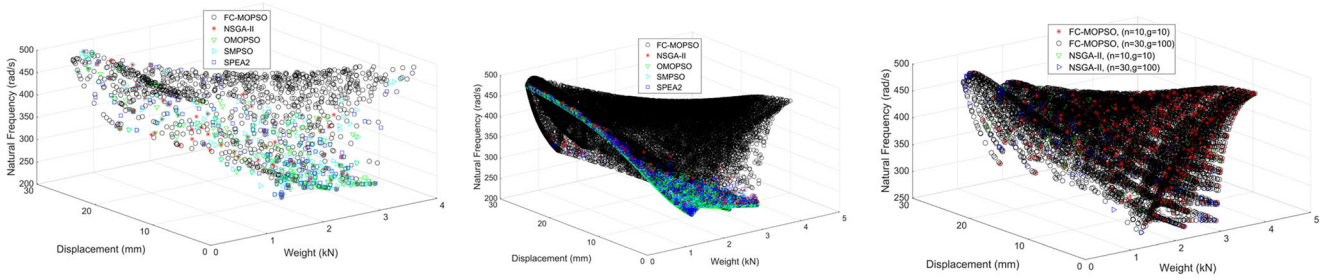
Table 6 List of equal leg double angles

Section No.		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Section ID.*		**2 L20 × 3**	**2 L25 × 3**	**2 L20 × 4**	**2 L30 × 3**	**2 L25 × 4**	**2 L35 × 3**	**2 L25 × 5**	**2 L30 × 4**
Cross sectional area	(mm^2)	224	284	290	348	370	408	452	454
	(in^2)	0.347	0.440	0.450	0.539	0.574	0.632	0.701	0.704
Section No.		**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Section ID.		**2 L40 × 3**	**2 L35 × 4**	**2 L30 × 5**	**2 L40 × 4**	**2 L35 × 5**	**2 L45 × 4**	**2 L40 × 5**	**2 L35 × 6**
Cross sectional area	(mm^2)	470	534	556	616	656	698	758	774
	(in^2)	0.729	0.828	0.862	0.955	1.017	1.082	1.175	1.200
Section No.		**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Section ID.		**2 L50 × 4**	**2 L45 × 5**	**2 L40 × 6**	**2 L50 × 5**	**2 L45 × 6**	**2 L55 × 5**	**2 L50 × 6**	**2 L60 × 5**
Cross sectional area	(mm^2)	778	860	896	960	1018	1064	1138	1164
	(in^2)	1.206	1.333	1.389	1.488	1.578	1.649	1.764	1.804
Section No.		**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**
Section ID.		**2 L45 × 7**	**2 L55 × 6**	**2 L50 × 7**	**2 L60 × 6**	**2 L50 × 8**	**2 L65 × 6**	**2 L70 × 6**	**2 L55 × 8**
Cross sectional area	(mm^2)	1172	1262	1312	1382	1482	1506	1626	1646
	(in^2)	1.817	1.956	2.034	2.142	2.297	2.334	2.520	2.551
Section No.		**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
Section ID.		**2 L50 × 9**	**2 L65 × 7**	**2 L75 × 6**	**2 L60 × 8**	**2 L70 × 7**	**2 L65 × 8**	**2 L55 × 10**	**2 L75 × 7**
Cross sectional area	(mm^2)	1648	1740	1750	1806	1880	1970	2020	2020
	(in^2)	2.554	2.697	2.713	2.799	2.914	3.054	3.131	3.131

*Equal leg angle sections are designated by $L \times t$ (mm), where L and t are respectively the leg size and the thickness of the leg

considered for simultaneous minimization: (1) the weight of the structure, (2) the maximum value of displacements of the uppermost nodes in X and Y directions, (3) the first natural frequency and (4) the total potential energy of the structure. The constraints imposed on the problem are to limit the stress of elements to ±25 *ksi*. The modulus of elasticity and density are assumed to be 10 *Msi* and 0.1 $lb./in^3$, respectively. Two load cases are considered. The first one is to apply P_X = 5.0 *kips*, P_Y = 5.0 *kips* and P_Z = −5.0 *kips* at node 17. The second load case consists of P_Z = −5 *kips*, applied to nodes 17, 18, 19 and 20. Cross-sectional areas of elements are linked into the following 16 groups: (1) $A_1 \sim A_4$, (2) $A_5 \sim A_{12}$, (3) $A_{13} \sim A_{16}$, (4) $A_{17} \sim A_{18}$, (5) $A_{19} \sim A_{22}$, (6) $A_{23} \sim A_{30}$, (7) $A_{31} \sim A_{34}$, (8) $A_{35} \sim A_{36}$, (9) $A_{37} \sim A_{40}$, (10) $A_{41} \sim A_{48}$, (11) $A_{49} \sim A_{52}$, (12) $A_{53} \sim A_{54}$, (13) $A_{55} \sim A_{58}$, (14) $A_{59} \sim A_{66}$, (15) $A_{67} \sim A_{70}$, and (16) $A_{71} \sim A_{72}$.

It is not possible to illustrate the 4D Pareto fronts obtained by the optimizers. Tables 7, and 8 show the extreme points of Pareto fronts found by different algorithms. However, they cannot be used for drawing a final conclusion since it can be observed that most extreme points of the Pareto fronts obtained from each optimizer is a nondominated point with respect to the corresponding extreme points found by other optimizers. Therefore, I_H and $I_{\varepsilon+}$ should be used for comparing the performance of the optimizers. Table 3 shows that the continuous FC-MOPSO outperforms other optimizers in terms of hypervolume. Standard deviations are also reasonably small and confirm the stability of FC-MOPSO. From Table 5, it can be confirmed that FC-MOPSO absolutely outperforms NSGA-II in discrete solutions. It is noted that the focus of FC-MOPSO is not solving problems with many objective functions (say, four and more



(a) Continuous solutions, n=10, g=10 (b) Continuous solutions, n=30, g=100 (c) Discrete solutions

Fig. 23 Comparisons of optimal solutions obtained by different optimizers for the 56 bar truss problem

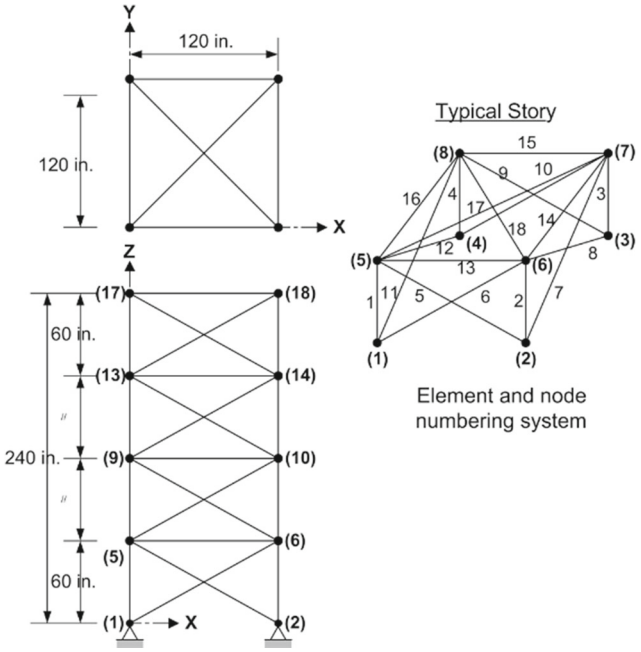


Fig. 24 Schematic of the spatial 72-bar truss structure

numbers of objective functions) since these kinds of problems are not conventional in design of civil structures.

6.2.4 Spatial 120-bar truss

The spatial 120-bar truss shown in Fig. 25 was defined by Soh and Yang (1996) and is taken here as an example of medium-scale structure to further analyze the performance of the FC-MOPSO algorithm. This truss was optimized for its weight by Lee and Geem (2004) by considering continuous tubular cross sections between 0.775 in^2 and 20.0 in^2. In this study, two objective functions are minimized: (1) the total weight of the structure (2) the maximum displacements of all nodes in all coordinate directions. Practical discrete tubular cross-sectional areas are defined in Table 9.

Furthermore, cross sections between 0.775 in^2 and 20.0 in^2 are considered for a continuous problem definition. Yield stress, modulus of elasticity and density are set equal to σ_y = 58 *ksi*, E = 30,450 *ksi* and ρ = 0.288 $lb./in^3$, respectively. Stress constraints based on AISC-ASD (1989) are applied, namely tensile stresses are limited to $\sigma^{-}_{all,i} = 0.6\sigma_y$ for each element while compressive stresses are limited to $\sigma^{+}_{all,i}$ as follows.

$$\sigma^{+}_{all,i} = \begin{cases} \left[\left(1-\frac{\lambda_i}{2C_c^2}\right)\sigma_y\right] \Big/ \left(\frac{5}{3}+\frac{3\lambda_i}{8C_c}-\frac{\lambda_i^3}{8C_c^3}\right), & if\ \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2}, & otherwise \end{cases} \tag{24}$$

where $C_c = (2\pi^2 E/\sigma_y)^{0.5}$ and $\lambda_i = k_i L_i/r_i$. k_i, L_i and r_i represent the effective length factor (equal to 1 for all members in this example), length of element and radius of gyration for each element, respectively. All elements are clustered into seven groups as defined in Fig. 25. All loads are applied in Z-direction and consist of −13.49 *kips* at node 1, −6.744 *kips* at nodes 2 through 14 and −2.247 *kips* on all other unsupported nodes.

Continuous solutions for 3000 function evaluations are presented in Fig. 26b. From this figure along with Table 3, it can be confirmed that all optimizers have yielded good approximations of the Pareto front. Especially, I_H and $I_{\varepsilon+}$ values for FC-MOPSO, SPEA2 and NSGA-II are very close to unity and zero, respectively. As it can be seen from Fig. 26a and Table 4, FC-MOPSO outperforms all other optimizers for 100 function evaluations. These results imply that the relative performance of FC-MOPSO, compared with other optimizers, is improved when a smaller number of function evaluations is used.

The best Pareto fronts for the discrete problem are shown in Fig. 26c from which the efficiency of FC- MOPSO over NSGA-II can be clearly confirmed. The superiority of FC-MOPSO is also confirmed from Table 5 where the $I_{\varepsilon+}$ indicators for FC-MOPSO are much better than those from NSGA-II. Hypervolume values for FC-MOPSO are very close to the values related to NSGA-II.

6.2.5 Sizing and geometric optimization of the spatial 25-bar truss with mixed design variables

The spatial 25-bar truss problem defined in Section 6.2.1 has been solved as a single-objective weight optimization problem with mixed design variables in the literature. See, for example, (Gholizadeh 2013) and (Mortazavi and Toğan 2016). In this paper, three objective functions (1) the weight of the structure (2) the maximum of all nodal displacements and (3) the first natural frequency of the structure will be minimized simultaneously. Material properties, load conditions, constraints on sizing variables and available discrete cross sectional areas are already defined in Section 6.2.1. The problem includes thirteen design variables: (1) A_1, (2) $A_2 \sim A_5$, (3) $A_6 \sim A_9$, (4) $A_{10} \sim A_{11}$, (5) $A_{12} \sim A_{13}$, (6) $A_{14} \sim A_{17}$, (7) $A_{18} \sim A_{21}$, (8) $A_{22} \sim A_{25}$ (9) $x_4 = x_5 = -x_3 = -x_6$, (10) $y_3 = y_4 = -y_5 = -y_6$, (11) $z_3 = z_4 = z_5 = z_6$, (12) $x_8 = x_9 = -x_7 = -x_{10}$, and (13) $y_7 = y_8 = -y_9 = -y_{10}$. The first eight variables are discrete sizing variables and the latter five variables represent nodal coordinates that are considered as continuous geometric variables. All other nodal coordinates are fixed at the coordinates shown in Fig. 20. Furthermore, the following constraints are applied on geometric variables:

$$20\ in. \le x_4 \le 60\ in.$$
$$40\ in. \le y_4 \le 80\ in.$$
$$90\ in. \le z_4 \le 130\ in.$$
$$40\ in. \le x_8 \le 80\ in.$$

Table 7 Extreme points of Pareto fronts found by different algorithms for the 72-bar truss problem with $n = 30$, $g = 100$

Algorithm	Vector of design variables* (in^2)	Vector of objective functions**
FC-MOPSO (Continuous)	[0.75, 0.10, 0.10, 2.50, 0.62, 0.21, 0.10, 0.87, 0.49, 0.10, 0.48, 1.18, 1.32, 0.35, 0.99, 0.93]	[424.43, 0.74, 103.55, 3593.69]
	[2.35, 2.44, 0.73, 2.50, 2.50, 2.50, 0.97, 1.28, 2.32, 2.50, 0.10, 0.10, 0.10, 2.50, 0.80, 0.66]	[1521.02, 0.08, 197.68, 1631.05]
	[0.39, 0.10, 1.36, 1.38, 1.39, 0.10, 2.35, 1.70, 2.14, 0.10, 1.74, 2.50, 0.21, 2.21, 1.17, 1.75]	[934.97, 0.83, 59.38, 4036.85]
	[2.50, 1.76, 2.04, 1.17, 2.13, 2.50, 1.23, 0.52, 1.92, 1.86, 0.29, 1.57, 2.50, 1.14, 1.35, 2.40]	[1423.45, 0.10, 189.79, 502.66]
OMOPSO (Continuous)	[2.45, 0.10, 0.62, 0.79, 1.06, 0.44, 0.10, 1.19, 1.55, 1.41, 1.97, 0.10, 1.33, 0.10, 0.10, 1.04]	[236.41, 0.92, 119.96, 4373.52]
	[2.50, 0.24, 0.69, 0.39, 2.38, 0.10, 2.50, 0.54, 2.50, 0.10, 0.65, 0.28, 2.32, 1.09, 2.16, 0.44]	[1706.86, 0.07, 175.95, 1478.85]
	[0.33, 0.10, 1.68, 2.50, 0.26, 0.10, 1.20, 0.19, 0.10, 2.50, 2.50, 2.50, 0.10, 2.50, 2.50, 2.50]	[1215.71, 0.76, 50.68, 3755.43]
	[2.50, 2.50, 2.47, 0.10, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 1.76, 2.50, 2.50, 2.50, 0.82, 2.50]	[1933.82, 0.08, 171.44, 438.37]
SMOPSO (Continuous)	[1.69, 0.26, 1.05, 1.35, 0.60, 2.30, 1.94, 1.07, 1.34, 0.92, 1.67, 1.76, 1.90, 1.22, 1.76, 1.51]	[297.58, 1.06, 124.76, 5270.81]
	[2.50, 2.50, 0.33, 2.50, 2.50, 2.50, 0.10, 1.89, 2.50, 2.34, 0.10, 0.10, 0.10, 2.40, 2.42, 2.50]	[1606.13, 0.07, 173.48, 1513.54]
	[0.87, 1.72, 2.16, 1.82, 0.92, 0.90, 2.15, 2.25, 1.79, 0.27, 0.15, 0.80, 1.80, 1.79, 0.90, 0.81]	[1215.71, 0.76, 50.68, 3755.43]
	[2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 0.10, 2.50]	[2017.52, 0.09, 172.78, 434.85]
NSGA-II (Continuous)	[1.62, 0.11, 0.50, 1.22, 0.27, 0.27, 0.65, 0.74, 0.88, 0.53, 0.75, 0.31, 0.92, 0.15, 0.15, 0.11]	[381.22, 0.71, 139.18, 3424.16]
	[0.29, 2.44, 1.68, 1.02, 2.47, 2.25, 1.71, 0.43, 2.36, 2.29, 0.16, 2.36, 0.89, 1.62, 1.96, 1.63]	[1669.86, 0.08, 174.31, 1160.71]
	[0.32, 0.10, 1.68, 0.98, 0.24, 0.10, 1.58, 1.53, 2.36, 2.22, 1.92, 2.45, 1.65, 2.43, 2.10, 1.64]	[1203.38, 0.75, 51.84, 3404.25]
	[0.32, 2.44, 1.68, 1.02, 2.47, 0.17, 1.58, 0.43, 1.92, 2.29, 0.16, 2.36, 2.06, 1.68, 1.96, 1.63]	[1646.40, 0.09, 168.73, 458.76]
SPEA2 (Continuous)	[1.11, 0.45, 0.76, 2.10, 2.01, 2.02, 0.78, 0.89, 1.03, 2.45, 2.43, 1.90, 0.46, 1.48, 0.87, 1.08]	[383.80, 0.49, 162.14, 2346.02]
	[1.64, 1.69, 0.91, 1.10, 0.46, 1.13, 0.91, 1.98, 2.44, 1.15, 2.14, 2.30, 2.10, 0.14, 0.81, 1.40]	[1669.86, 0.08, 174.31, 1160.71]
	[1.16, 0.49, 1.39, 0.60, 1.73, 1.45, 0.34, 0.30, 0.75, 1.27, 2.21, 0.80, 0.41, 0.15, 2.02, 1.40]	[1203.38, 0.75, 51.84, 3404.25]
	[0.94, 1.76, 1.26, 0.29, 2.40, 0.16, 0.90, 1.68, 2.04, 2.12, 0.67, 0.29, 1.63, 1.34, 0.62, 0.30]	[1813.61, 0.09, 162.26, 457.06]
FC-MOPSO (Discrete)	[7, 3, 1, 2, 24, 7, 19, 4, 2, 6, 2, 2, 7, 3, 2, 15]	[540.71, 0.29, 166.71, 1545.91]
	[30, 25, 5, 9, 31, 30, 14, 2, 22, 31, 7, 24, 4, 31, 14, 31]	[1507.04, 0.08, 174.08, 790.60]
	[1, 1, 6, 29, 1, 1, 16, 25, 9, 26, 29, 25, 2, 31, 17, 26]	[1123.46, 0.41, 75.87, 2260.34]
	[31, 23, 5, 16, 30, 28, 16, 1, 28, 31, 3, 18, 30, 14, 15, 14]	[1327.36, 0.10, 200.01, 490.31]
NSGA-II (Discrete)	[1, 1, 3, 16, 11, 8, 9, 9, 3, 2, 8, 8, 3, 7, 11, 8]	[530.73, 0.40, 129.12, 2236.35]
	[30, 31, 14, 9, 27, 24, 25, 25, 25, 29, 18, 11, 18, 30, 29, 23]	[1630.55, 0.09, 168.55, 601.00]
	[1, 7, 10, 3, 8, 1, 3, 15, 25, 28, 25, 16, 15, 21, 26, 15]	[986.88, 0.33, 92.11, 1453.57]
	[27, 26, 15, 24, 26, 31, 27, 27, 29, 27, 27, 30, 29, 22, 19, 17]	[1650.45, 0.10, 170.57, 512.98]

*Integer numbers reported for optimized design variables of discrete solutions refer to section numbers as defined in Table 6

**The order of vector of objective functions is [Weight (*lb*), Displacement (*in*), Natural frequency (*rad*/*s*), Potential energy (*lb*-*in*)]

Table 8 Extreme points of Pareto fronts found by different algorithms for the 72-bar truss problem with $n = 10$, $g = 10$

Algorithm	Vector of design variables* (in^2)	Vector of objective functions**
FC-MOPSO (Continuous)	[1.29, 0.10, 1.42, 0.77, 1.50, 0.47, 0.88, 0.92, 0.57, 0.10, 1.40, 0.80, 0.17, 0.60, 0.10, 2.48]	[572.28, 0.60, 101.63, 3107.04]
	[2.49, 2.17, 1.12, 2.41, 1.80, 2.18, 1.29, 2.05, 1.23, 1.66, 0.32, 0.26, 1.92, 1.06, 1.35, 1.97]	[1360.31, 0.11, 198.33, 639.75]
	[0.37, 0.10, 2.50, 0.40, 1.84, 0.10, 2.50, 1.51, 0.10, 2.50, 1.37, 2.02, 1.63, 0.56, 1.60, 1.90]	[1025.29, 0.69, 62.50, 3238.67]
	[2.50, 1.46, 1.64, 0.43, 2.04, 1.92, 1.67, 0.81, 1.90, 1.49, 0.68, 2.26, 1.85, 1.23, 0.48, 0.55]	[1206.06, 0.12, 203.52, 579.11]
OMOPSO (Continuous)	[2.18, 0.10, 0.10, 0.10, 1.69, 0.10, 0.10, 1.12, 1.00, 0.10, 2.50, 2.50, 2.38, 0.10, 0.28, 2.24]	[562.25, 0.92, 80.80, 4604.27]
	[1.79, 2.50, 2.48, 0.30, 2.11, 2.50, 2.19, 1.69, 2.50, 2.50, 2.32, 0.64, 1.05, 2.35, 2.50, 1.02]	[1815.46, 0.09, 162.17, 617.78]
	[0.40, 0.10, 1.07, 0.10, 0.10, 2.14, 2.50, 1.86, 2.50, 0.10, 2.09, 2.50, 2.05, 2.50, 1.18, 0.10]	[1123.73, 0.78, 60.40, 3462.46]
	[2.50, 1.84, 0.10, 2.50, 2.03, 2.50, 0.10, 2.50, 2.50, 2.50, 0.10, 1.04, 2.50, 2.22, 0.36, 2.50]	[1522.99, 0.10, 181.59, 477.78]
SMOPSO (Continuous)	[2.50, 0.22, 2.50, 0.10, 2.41, 0.26, 0.10, 0.84, 0.67, 0.22, 1.08, 0.10, 0.97, 0.10, 1.83, 0.56]	[561.96, 0.55, 135.30, 2779.74]
	[1.77, 1.97, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 2.50, 0.10, 0.63, 2.50, 2.50, 2.50]	[1931.75, 0.09, 153.12, 700.77]
	[1.55, 0.10, 0.23, 1.08, 0.30, 0.10, 1.00, 0.92, 0.10, 2.48, 0.56, 2.03, 2.50, 1.18, 2.30, 1.59]	[908.03, 0.62, 65.30, 2905.59]
	[2.50, 1.16, 2.01, 2.50, 2.50, 2.50, 2.50, 1.20, 2.50, 2.50, 2.50, 2.50, 1.24, 2.50, 0.10, 2.50]	[1776.18, 0.10, 160.97, 534.10]
NSGA-II (Continuous)	[1.20, 0.38, 0.11, 1.44, 1.90, 0.55, 0.43, 2.45, 0.90, 0.11, 0.61, 1.08, 0.28, 0.45, 0.90, 1.28]	[573.09, 0.41, 141.73, 2138.74]
	[2.43, 1.59, 1.92, 0.68, 2.30, 2.36, 2.00, 1.97, 2.13, 1.94, 1.26, 1.89, 0.40, 2.40, 1.17, 1.74]	[1581.85, 0.09, 173.87, 829.35]
	[0.95, 0.10, 1.29, 1.99, 0.56, 1.66, 0.98, 0.28, 1.02, 2.16, 1.83, 0.66, 0.73, 2.40, 1.59, 1.42]	[1176.78, 0.37, 76.86, 1737.35]
	[2.31, 1.52, 1.63, 1.99, 2.39, 2.40, 2.40, 0.49, 2.06, 1.72, 1.85, 2.04, 1.84, 1.86, 1.07, 1.69]	[1555.74, 0.10, 174.47, 517.02]
SPEA2 (Continuous)	[1.85, 0.19, 0.92, 1.80, 0.48, 0.23, 0.85, 1.10, 0.52, 0.20, 1.14, 1.04, 0.94, 0.25, 1.16, 2.11]	[584.85, 0.50, 113.27, 2483.05]
	[2.07, 2.42, 1.04, 1.30, 1.79, 1.44, 1.28, 0.50, 1.66, 2.36, 1.16, 1.93, 0.33, 2.42, 1.09, 2.48]	[1498.92, 0.09, 164.95, 966.34]
	[0.22, 2.47, 1.14, 2.24, 1.09, 0.12, 0.94, 2.45, 2.05, 2.08, 1.72, 1.64, 1.83, 2.42, 1.12, 0.33]	[1346.96, 0.44, 76.21, 1939.18]
	[2.12, 1.80, 1.63, 0.87, 2.00, 2.47, 2.16, 1.35, 2.25, 1.52, 0.24, 2.43, 2.11, 1.32, 0.82, 2.48]	[1441.90, 0.11, 182.85, 524.62]
FC-MOPSO (Discrete)	[11, 3, 2, 10, 16, 8, 17, 5, 7, 3, 13, 14, 26, 6, 8, 5]	[618.48, 0.29, 152.18, 1336.79]
	[31, 27, 30, 12, 27, 28, 30, 13, 18, 21, 19, 16, 3, 27, 16, 17]	[1484.79, 0.09, 187.17, 908.84]
	[1, 3, 2, 25, 2, 14, 22, 10, 7, 8, 23, 14, 5, 26, 31, 19]	[978.87, 0.36, 87.95, 2028.85]
	[31, 29, 24, 11, 28, 27, 17, 16, 31, 25, 12, 25, 24, 17, 7, 23]	[1420.36, 0.10, 201.26, 505.94]
NSGA-II (Discrete)	[6, 10, 4, 15, 6, 3, 10, 2, 1, 11, 4, 18, 17, 2, 8, 18]	[615.05, 0.32, 144.32, 1764.41]
	[23, 31, 16, 16, 20, 20, 23, 5, 3, 23, 22, 16, 23, 28, 31, 26]	[1490.31, 0.11, 145.40, 965.07]
	[2, 3, 28, 11, 7, 22, 28, 9, 6, 5, 11, 18, 23, 30, 30, 25]	[1142.18, 0.29, 93.37, 1510.79]
	[24, 27, 24, 9, 27, 19, 10, 8, 23, 7, 19, 20, 25, 29, 21, 8]	[1258.85, 0.13, 164.33, 598.60]

*Integer numbers reported for optimized design variables discrete solutions refer to section numbers as defined in Table 6

**The order of vector of objective functions is [Weight (*lb*), Displacement (*in*), Natural frequency (*rad*/*s*), Potential energy (*lb*-*in*)

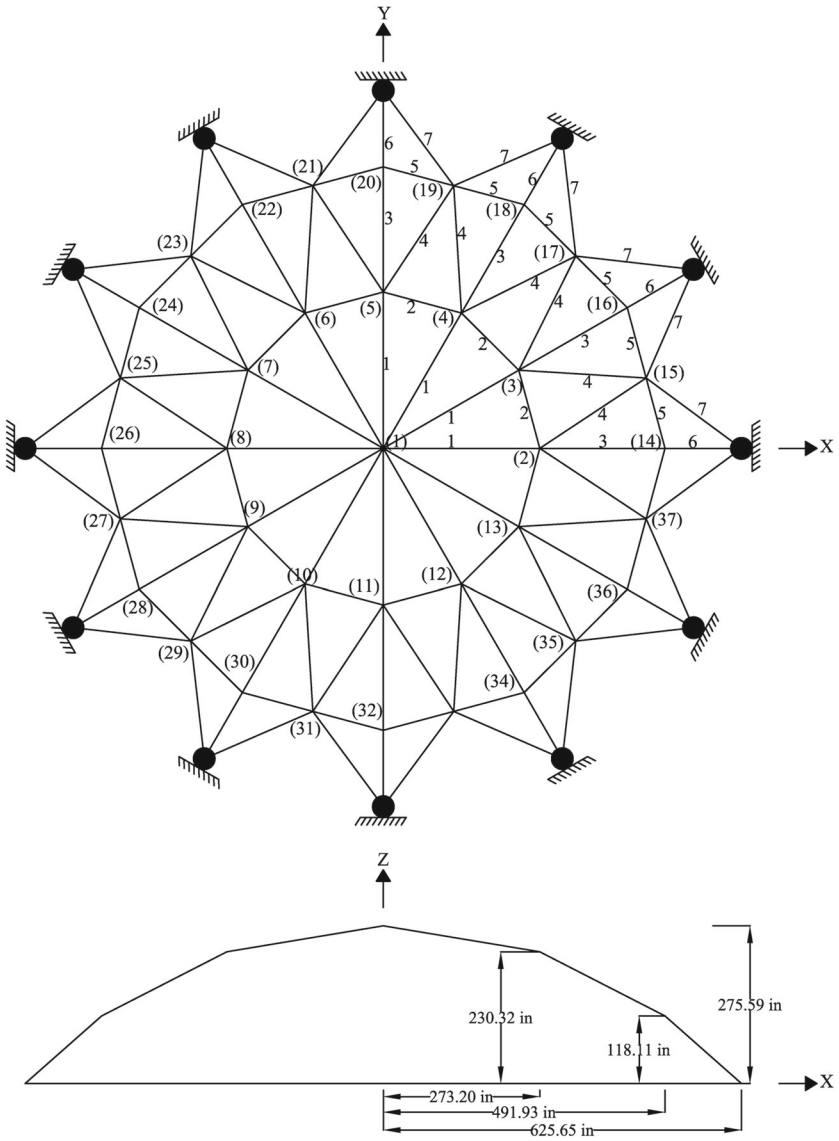


Fig. 25 Schematic of the spatial 120-bar truss structure

$$100\ in. \le y_8 \le 140\ in.$$

Continuous and discrete FC-MOPSO should be utilized simultaneously for solving the problem. In other words, the algorithm of Section 4 is used for dealing with continuous geometric variables while the algorithm proposed in Section 5 is used for dealing with discrete sizing variables. This method shall be called mixed FC-MOPSO.

The results obtained from FC-MOPSO and NSGA-II are illustrated in Fig. 27. An approximation of true Pareto front is also illustrated. The true Pareto front was approximated by considering all Pareto fronts that were obtained from all simulations of the two optimizers.

Fig. 27a and b show that FC-MOPSO clearly outperforms NSGA-II in solving the problem with mixed continuous and discrete design variables. In fact, NSGA-II could approximate a small portion of the Pareto front. Fig. 27b shows that almost all points corresponding to the approximation of true Pareto belong to FC-MOPSO.

Extreme points of the Pareto fronts are also listed in Table 10. However, as it was discussed in Section 6.2.3, this table cannot be used for preferring one algorithm over another.

7 Conclusions

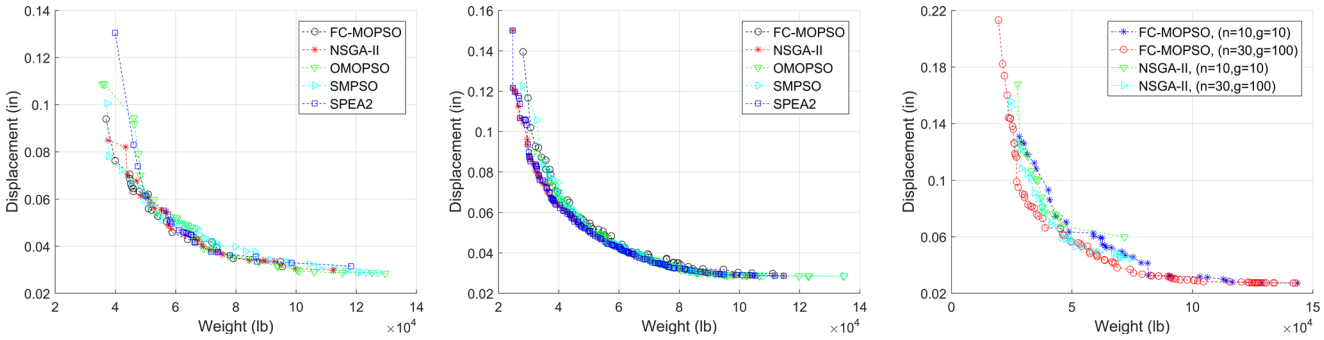
A new PSO-based approach called FC-MOPSO was developed for handling multi-objective optimization problems with continuous or discrete variables and including different types of constraints. The extension of the original single-objective PSO to a multi-objective PSO was carefully accomplished to preserve the basic rationale in PSO.

FC-MOPSO reserves the best positions that each individual has experienced as a nondominated set. These non-dominated sets are

Table 9 List of tubular sections employed for the 120-bar truss example

Section No.		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Round tube ID.*		**48.3 × 3.2**	**42.4 × 4**	**60.3 × 2.9**	**48.3 × 4**	**60.3 × 3.6**	**76.1 × 2.9**	**60.3 × 4**	**76.1 × 3.6**
Cross sectional area	(mm^2)	453	483	523	557	641	667	707	820
	(in^2)	0.702	0.749	0.811	0.863	0.994	1.034	1.096	1.271
Radius of gyration	(*mm*)	16	13.6	20.3	15.7	20.1	25.9	20	25.7
	(*in*)	0.630	0.535	0.799	0.618	0.791	1.020	0.787	1.012
Section No.		**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Round tube ID.		**88.9 × 3.2**	**60.3 × 5**	**76.1 × 4**	**88.9 × 3.6**	**88.9 × 4**	**101.6 × 3.6**	**76.1 × 5**	**108 × 3.6**
Cross sectional area	(mm^2)	862	869	906	965	1070	1110	1120	1180
	(in^2)	1.336	1.347	1.404	1.496	1.659	1.721	1.736	1.829
Radius of gyration	(*mm*)	30.3	19.6	25.5	30.2	30	34.7	25.2	36.9
	(*in*)	1.193	0.772	1.004	1.189	1.181	1.366	0.992	1.453
Section No.		**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Round tube ID.		**114.3 × 3.6**	**88.9 × 5**	**101.6 × 4.5**	**108 × 4.5**	**114.3 × 4.5**	**133 × 4**	**88.9 × 6.3**	**101.6 × 5.6**
Cross sectional area	(mm^2)	1250	1320	1370	1460	1550	1620	1630	1690
	(in^2)	1.938	2.046	2.124	2.263	2.403	2.511	2.527	2.62
Radius of gyration	(*mm*)	39.2	29.7	34.4	36.6	38.9	45.6	29.3	34
	(*in*)	1.543	1.169	1.354	1.441	1.531	1.795	1.154	1.339
Section No.		**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**
Round tube ID.		**139.7 × 4**	**108 × 5.6**	**114.3 × 5.6**	**101.6 × 7.1**	**159 × 4.5**	**133 × 5.6**	**168.3 × 4.5**	**139.7 × 5.6**
Cross sectional area	(mm^2)	1710	1800	1910	2110	2180	2240	2320	2360
	(in^2)	2.651	2.79	2.961	3.271	3.379	3.472	3.596	3.658
Radius of gyration	(*mm*)	48	36.3	38.5	33.5	54.6	45.1	57.9	47.5
	(*in*)	1.890	1.429	1.516	1.319	2.150	1.776	2.280	1.870
Section No.		**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
Round tube ID.		**114.3 × 7.1**	**133 × 6.3**	**139.7 × 6.3**	**193.7 × 4.5**	**159 × 5.6**	**168.3 × 5.6**	**139.7 × 7.1**	**193.7 × 5**
Cross sectional area	(mm^2)	2390	2510	2640	2670	2700	2860	2958	2964
	(in^2)	3.705	3.891	4.092	4.139	4.185	4.433	4.584	4.594
Radius of gyration	(*mm*)	38	44.9	47.2	66.9	54.3	57.6	46.9	66.7
	(*in*)	1.496	1.768	1.858	2.634	2.138	2.268	1.846	2.626
Section No.		**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**
Round tube ID.		**159 × 6.3**	**219.1 × 4.5**	**193.7 × 5.4**	**168.3 × 6.3**	**219.1 × 5**	**168.3 × 7.1**	**193.7 × 6.3**	**219.1 × 5.9**
Cross sectional area	(mm^2)	3020	3030	3190	3210	3360	3600	3710	3950
	(in^2)	4.681	4.697	4.945	4.976	5.208	5.58	5.751	6.123
Radius of gyration	(*mm*)	54	75.9	66.6	57.3	75.7	57	66.3	75.4
	(*in*)	2.126	2.988	2.622	2.256	2.980	2.244	2.610	2.969
Section No.		**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**
Round tube ID.		**193.7 × 7.1**	**273 × 5**	**219.1 × 6.3**	**168.3 × 8.8**	**193.7 × 8**	**273 × 5.6**	**244.5 × 6.3**	**219.1 × 7.1**
Cross sectional area	(mm^2)	4160	4209.734	4211.745	4410	4670	4700	4710	4730
	(in^2)	6.448	6.525	6.528	6.836	7.239	7.285	7.301	7.332
Radius of gyration	(*mm*)	66	94.8	75.3	56.5	65.7	94.6	84.2	75
	(*in*)	2.598	3.732	2.965	2.224	2.587	3.724	3.315	2.953
Section No.		**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**
Round tube ID.		**323.9 × 5**	**193.7 × 8.8**	**273 × 6.3**	**323.9 × 5.6**	**219.1 × 8.8**	**273 × 7.1**	**244.5 × 8**	**323.9 × 6.3**
Cross sectional area	(mm^2)	5010	5110	5280	5600	5810	5930	5940	6290
	(in^2)	7.766	7.921	8.184	8.68	9.006	9.192	9.207	9.75
Radius of gyration	(*mm*)	113	65.4	94.3	113	74.4	94	83.7	112
	(*in*)	4.449	2.575	3.713	4.449	2.929	3.701	3.295	4.409
Section No.		**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**
Round tube ID.		**219.1 × 10**	**323.9 × 7.1**	**273 × 8.8**	**244.5 × 10**	**244.5 × 11**	**323.9 × 8.8**	**355.6 × 8**	**273 × 11**
Cross sectional area	(mm^2)	6570	7070	7300	7330	8070	8710	8740	9050
	(in^2)	10.184	10.959	11.315	11.362	12.509	13.501	13.547	14.028
Radius of gyration	(*mm*)	74	112	93.7	83	82.6	111	123	92.7
	(*in*)	2.913	4.409	3.689	3.268	3.252	4.370	4.843	3.650
Section No.		**73**	**74**	**75**	**76**				
Round tube ID.		**323.9 × 11**	**355.6 × 10**	**406.4 × 8.8**	**355.6 × 12.5**				
Cross sectional area	(mm^2)	10,800	10,900	11,000	13,500				
	(in^2)	16.74	16.895	17.05	20.925				
Radius of gyration	(*mm*)	111	122	141	121				
	(*in*)	4.370	4.803	5.551	4.764				

*Tubular sections are designated by $D \times t$ (*mm*), where D and t are respectively the diameter and the thickness of a tubular section



(a) Continuous solutions (n=10, g=10) (b) Continuous solutions (n=30, g=100) (c) Discrete solutions

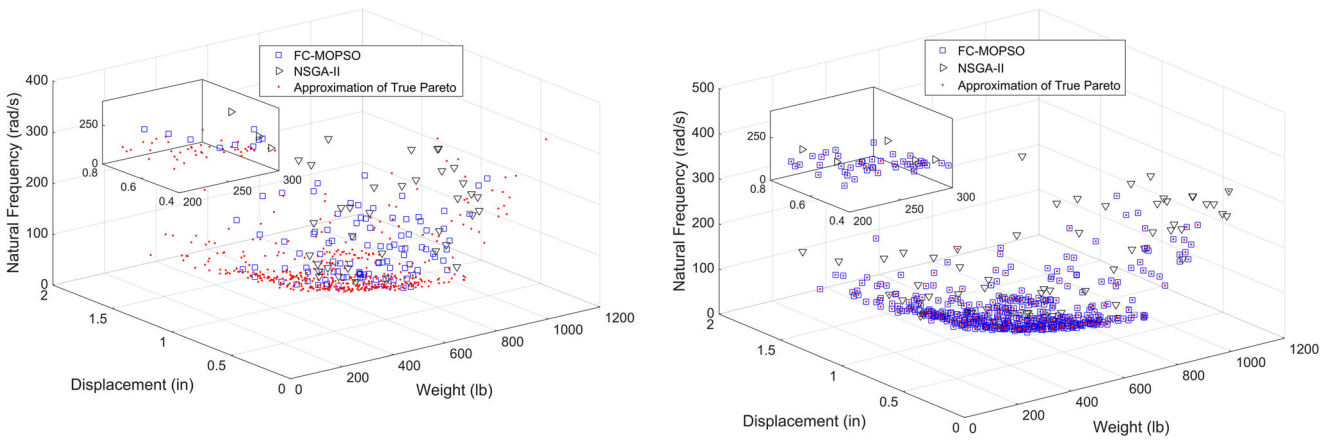
Fig. 26 Comparisons of optimal solutions obtained by different optimizers for the 120 bar truss problem

subsequently used for selecting a leader and a local best particle for each individual and updating its position. This feature supports the swarm with the most possible information about the best regions of the search space. Such procedure can result in a fast rate of convergence but may also lead to converging to local minima. However, FC-MOPSO avoids premature convergence by restricting the size of archives, utilizing appropriate selection procedures and applying mutation operators. Restricting the size of archives and the selection procedure is done in a way that the swarm is encouraged to move to less explored regions of the search space. Such characteristics increase the chance of finding an acceptable approximation of the Pareto in limited number of function evaluations. Furthermore, extension of the single-objective PSO to FC-MOPSO was accomplished in a way that it is also possible to use any desired neighborhood topology other than the random one that was recommended in this paper. In other words, any neighborhood topology defined for single objective PSO methods can be readily applied to FC-MOPSO as well.

The methodologies proposed in this paper are effective, consistent with each other and easy to implement. The continuous and discrete FC-MOPSO can be readily mixed with each other to solve problems with mixed design variables. Therefore, FC-MOPSO can be applied to unconstrained/constrained continuous/ discrete/ mixed multi-objective problems while most other PSO-based algorithms available in the literature are not meant to address all these scenarios.

The performance of FC-MOPSO was validated by considering a wide range of well-known benchmark problems. It was shown that the proposed method performs very well in finding acceptable approximations of Pareto fronts within limited number of function evaluations. This characteristic along with the capability of handling discrete and mixed search spaces makes FC-MOPSO an appropriate tool for structural optimization problems. It is an effective method because most structural MOPs deal with discrete or mixed discrete and continuous design variables on one hand and suffer from high costs in function evaluations on the other hand. Function evaluations can be quite costly especially for cases where more rigorous analyses such as nonlinear time history analyses are to be adopted.

The proposed method avoids defining parameters that need to be tuned. Two types of parameters could be distinguished in FC-



(a) n=10, g=10 (b) n=30, g=100

Fig. 27 Comparisons of optimal solutions obtained by FC-MOPSO and NSGA-II for sizing and geometric optimization of the 25 bar truss problem

Table 10 Extreme points of Pareto fronts for sizing and geometric optimization of the 25-bar truss problem

Number of function evaluations	Algorithm	Vector of design variables*	Vector of objective functions**
100 (n = 10, g = 10)	FC-MOPSO	[1.7, 1, 0.4, 1.6, 0.1, 0.1, 0.5, 0.5, 20.00, 43.65, 110.16, 66.74, 116.48]	[171.42, 0.94, 180.34]
		[3.4, 3.4, 3.4, 2.6, 0.5, 3.4, 1.2, 3.2, 22.56, 61.01, 130.00, 80.00, 117.90]	[962.50, 0.10, 288.49]
		[2.1, 2.1, 2, 0.4, 0.1, 0.2, 1.4, 0.1, 20.82, 49.85, 122.78, 40.50, 127.47]	[302.21, 1.04, 67.30]
	NSGA-II	[0.5, 0.2, 1.4, 0.7, 0.6, 0.2, 0.5, 0.2, 36.36, 55.31, 122.08, 71.72, 133.65]	[157.48, 0.72, 205.68]
		[0.9, 2.1, 3.4, 0.6, 1.8, 1.9, 1.2, 3.2, 39.50, 63.01, 119.52, 59.55, 122.63]	[718.04, 0.10, 289.62]
		[2.1, 1.5, 1.6, 0.4, 1, 2.8, 0.7, 0.1, 20.26, 68.58, 95.96, 47.15, 113.98]	[470.28, 0.78, 83.72]
3000 (n = 30, g = 100)	FC-MOPSO	[0.2, 0.3, 0.5, 0.4, 0.8, 0.3, 0.4, 0.3, 39.79, 45.00, 118.72, 52.96, 128.26]	[124.78, 1.01, 252.21]
		[1.8, 3.4, 3.4, 1.9, 1, 3.2, 3, 2.6, 32.95, 62.43, 122.01, 57.06, 140.00]	[1037.29, 0.09, 259.64]
		[0.1, 3, 3.4, 0.6, 1.6, 0.4, 1.6, 0.1, 20.00, 40.23, 90.00, 40.00, 140.00]	[472.84, 1.80, 44.99]
	NSGA-II	[0.3, 0.5, 0.2, 0.5, 0.2, 0.2, 0.2, 0.3, 30.27, 65.50, 107.68, 62.65, 116.27]	[97.22, 1.04, 272.15]
		[3, 2, 3.4, 1.2, 2.2, 2, 2.8, 3.2, 44.36, 65.61, 116.77, 65.86, 133.87]	[901.02, 0.09, 332.87]
		[2.3, 3, 3, 3, 0.3, 3, 0.1, 2.1, 57.81, 47.06, 114.20, 43.41, 109.14]	[713.81, 0.50, 64.09]

* Sizing and geometric variables are reported in in^2 and in, respectively

**The order of vector of objective functions is [Weight (lb), Displacement (in), Natural frequency (rad/s)]

MOPSO: (1) the three parameters inherited from its ancestor (i.e. ϕ_1, ϕ_2 and α) and (2) all other parameters specifically defined for FC-MOPSO such as p_m, ρ_j and $C(x_i)$. The user is not asked to tune type 1 parameters because of the theoretical background for stability of PSO algorithms. Type 2 parameters are not required to be tuned either. Because, they were defined in a way that parameter tuning would not be required.

The number of design variables of the problems considered in this paper was at most sixteen variables for truss examples and thirty variables for continuous benchmark MOPs. FC-MOPSO could find satisfactory results for these problems within 100 and 400 function evaluations. Naturally, for larger problems it may be necessary to use higher number of function evaluations. It can be done by considering larger populations or larger number of generations or both. If the population size is very small (say for example, n = 10), the swarm will receive little information about the search space. Moreover, it is likely that the number of function evaluations is small in such cases (for example, large number of evaluations like n = 10, g = 1000 is an unlikely scenario because n = 100, g = 100 is probably preferred) and the swarm will have to converge to a solution more rapidly. Therefore, using neighborhood sizes equal to n - 1 is recommended in such cases. N = n - 1 somehow represents a fully informed PSO. For larger population sizes it is recommended to adopt small neighborhood sizes to restrict the amount of information passed to each particle and thereby, avoiding premature convergence. For example, N = 4 and N = 8 can be used for n = 30 and n = 100, respectively.

References

Adeli H, Kamal O (1986) Efficient optimization of space trusses. Comput Struct 24:501–511. doi:10.1016/0045-7949(86)90327-5

AISC A (1989) Manual of steel construction-allowable stress design. American Institute of Steel Construction (AISC), Chicago

Chafekar D, Shi L, Rasheed K, Xuan J (2005) Multiobjective GA optimization using reduced models. IEEE Trans Syst Man Cybern Part C Appl Rev 35:261–265. doi:10.1109/TSMCC.2004.841905

Chen J, Chen G, Guo W (2009) A discrete PSO for multi-objective optimization in VLSI floorplanning. In: Cai Z, Li Z, Kang Z, Liu Y (eds) Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). Springer, Berlin Heidelberg, pp 400–410

Chowdhury S, Tong W, Messac A, Zhang J (2013) A mixed-discrete particle swarm optimization algorithm with explicit diversity-preservation. Struct Multidiscip Optim 47:367–388. doi:10.1007/s00158-012-0851-z

Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6:58–73. doi:10.1109/4235.985692

Coello Coello CA, Lechuga MS (2002) MOPSO: A proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002. IEEE, pp 1051–1056

Coello Coello C, Lamont GB, van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems. Springer US, Boston

Davarynejad M, Rezaei J, Vrancken J, et al (2011) Accelerating convergence towards the optimal pareto front. In: 2011 I.E. Congress of Evolutionary Computation, CEC 2011. IEEE, pp 2107–2114

Deb K, Deb D (2014) Analysing mutation schemes for real-parameter genetic algorithms. Int J Artif Intell Soft Comput 4:1. doi:10.1504/IJAISC.2014.059280

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197. doi:10.1109/4235.996017

Deb K, Pratap A, Meyarivan T (2001) Constrained test problems for multi-objective evolutionary optimization. In: Zitzler E, Thiele L, Deb K et al (eds) Evolutionary multi-criterion optimization. Springer, Berlin Heidelberg, pp 284–298

Durillo JJ, Nebro AJ (2011) jMetal: A Java framework for multi-objective optimization. Adv Eng Softw 42:760–771. doi:10.1016/j.advengsoft.2011.05.014

Erbatur F, Hasançebi O, Tütüncü İ, Kılıç H (2000) Optimal design of planar and space structures with genetic algorithms. Comput Struct 75:209–224. doi:10.1016/S0045-7949(99)00084-X

Eskandari H, Geiger CD (2008) A fast Pareto genetic algorithm approach for solving expensive multiobjective optimization problems. J Heuristics 14:203–241. doi:10.1007/s10732-007-9037-z

Fonseca CM, Knowles JD, Thiele L, Zitzler E (2005) A tutorial on the performance assessment of stochastic multiobjective optimizers. In: Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005). p 240

Gholizadeh S (2013) Layout optimization of truss structures by hybridizing cellular automata and particle swarm optimization. Comput Struct 125:86–99. doi:10.1016/j.compstruc.2013.04.024

Goldberg DE, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp 41–49

Golinski J (1970) Optimal synthesis problems solved by means of nonlinear programming and random methods. J Mech 5:287–309. doi:10.1016/0022-2569(70)90064-9

Gong Y, Xue Y, Xu L (2013) Optimal capacity design of eccentrically braced steel frameworks using nonlinear response history analysis. Eng Struct 48:28–36. doi:10.1016/j.engstruct.2012.10.001

Jansen PW, Perez RE (2011) Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach. Comput Struct 89:1352–1366. doi:10.1016/j.compstruc.2011.03.011

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. IEEE, pp 1942–1948

Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: 1997 I.E. International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. IEEE, pp 4104–4108

Kurpati A, Azarm S, Wu J (2002) Constraint handling improvements for multiobjective genetic algorithms. Struct Multidiscip Optim 23:204–213. doi:10.1007/s00158-002-0178-2

Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798. doi:10.1016/j.compstruc.2004.01.002

Luh GC, Chueh CH (2004) Multi-objective optimal design of truss structure with immune algorithm. Comput Struct 82:829–844. doi:10.1016/j.compstruc.2004.03.003

Mortazavi A, Toğan V (2016) Simultaneous size, shape, and topology optimization of truss structures using integrated particle swarm optimizer. Struct Multidiscip Optim 54:715–736. doi:10.1007/s00158-016-1449-7

Nebro AJ, Durillo JJ, Nieto G, et al (2009) SMPSO: A new pso-based metaheuristic for multi-objective optimization. In: 2009 I.E. Symposium on Computational Intelligence in Multi-Criteria Decision-Making, MCDM 2009 - Proceedings. IEEE, pp 66–73

Paya I, Yepes V, González-Vidosa F, Hospitaler A (2008) Multiobjective optimization of concrete frames by simulated annealing. Comput Civ Infrastruct Eng 23:596–610. doi:10.1111/j.1467-8667.2008.00561.x

Ray T, Kang T, Kian Chye S (2001) Multiobjective design optimization by an evolutionary algorithm. Eng Optim 33:399–424. doi:10.1080/03052150108940926

Reyes-Sierra M, Coello Coello CA (2006) Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int J Comput Intell Res 2:287–308

Santana-Quintero LV, Montaño AA, Coello Coello CA (2010) A review of techniques for handling expensive functions in evolutionary multi-objective optimization. In: Tenne Y, Goh C-K (eds) Computational intelligence in expensive optimization problems. Springer, Berlin Heidelberg, pp 29–59

Sierra MR, Coello Coello CA (2005) Improving PSO-based multi-objective optimization using crowding, mutation and ∈−dominance. In: Coello Coello CA, Hernández Aguirre A, Zitzler E (eds) Evolutionary multi-criterion optimization: Third International Conference, EMO 2005, Guanajuato, Mexico, March 9–11, 2005, Proceedings. Springer, Berlin Heidelberg, pp 505–519

Simon D (2013) Evolutionary optimization algorithms. John Wiley & Sons, Hoboken

Soh CK, Yang J (1996) Fuzzy controlled genetic algorithm search for shape optimization. J Comput Civ Eng 10:143–150. doi:10.1061/(ASCE)0887-3801(1996)10:2(143)

Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2:221–248. doi:10.1162/evco.1994.2.3.221

Talatahari S, Kaveh A, Sheikholeslami R (2012) Chaotic imperialist competitive algorithm for optimum design of truss structures. Struct Multidiscip Optim 46:355–367. doi:10.1007/s00158-011-0754-4

Tanaka M, Watanabe H, Furukawa Y, Tanino T (1995) GA-based decision support system for multicriteria optimization. In: IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century. IEEE, pp 1556–1561 vol. 2

Tong W, Chowdhury S, Messac A (2016) A multi-objective mixed-discrete particle swarm optimization with multi-domain diversity preservation. Struct Multidiscip Optim 53:471–488. doi:10.1007/s00158-015-1319-8

Yan J, Li C, Wang Z, et al (2007) Diversity metrics in multi-objective optimization: Review and perspective. In: IEEE ICIT 2007–2007 I.E. International Conference on Integration Technology. IEEE, pp 553–557

Yen GG (2009) An adaptive penalty function for handling constraint in multi-objective evolutionary optimization. In: Mezura-Montes E (ed) Constraint-handling in evolutionary optimization. Springer, Berlin, Heidelberg, pp 121–143

Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8:173–195. doi:10.1162/106365600568202

Zitzler E, Laumanns M, Thiele L (2002) SPEA2: Improving the strength pareto evolutionary algorithm. In: Giannakoglou K, Tsahalis D, Periaux P, et al (eds) Evolutionary methods for design, optimization and control with applications to industrial problems. CIMNE, Barcelona, pp 95–100