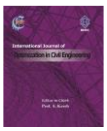
INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng.,2018; 8(1):53-75



A FAST FUZZY-TUNED MULTI-OBJECTIVE OPTIMIZATION FOR SIZING PROBLEMS

M. Shahrouzi[*, †] and H. Farah-Abadi
Faculty of Engineering, Kharazmi University, Tehran, Iran

ABSTRACT

The most recent approaches of multi-objective optimization constitute application of meta-heuristic algorithms for which, parameter tuning is still a challenge. The present work hybridizes swarm intelligence with fuzzy operators to extend crisp values of the main control parameters into especial fuzzy sets that are constructed based on a number of prescribed facts. Such parameter-less particle swarm optimization is employed as the core of a multi-objective optimization framework with a repository to save Pareto solutions. The proposed method is tested on a variety of benchmark functions and structural sizing examples. Results show that it can provide Pareto front by lower computational time in competition with some other popular multi-objective algorithms.

Keywords: fuzzy logic; parameter reduction; multi-objective optimization; swarm intelligence; Pareto front.

Received: 2 April 2017; Accepted: 26 June 2017

1. INTRODUCTION

Many real-world engineering problems fall in the category of Pareto optimization due to existence of conflicting objectives. It means an objective cannot be improved without deterioration of at least another objective; therefore these problems have more than one solution; that is called Pareto front, PF.

Up to date, several methods have been offered by investigators for multi-objective optimization, MOO. A wide variety of them includes evolutionary methods that generally constitute improvement of both the algorithms and the data structures to extract non-dominated solutions. This category is historically considered a revolution in their time; that includes VEGA [1], MOGA [2], NPGA [3], SPEA-II [4]. Particularly, Non-dominated Sorting Genetic Algorithm-II [5] has received much attention. NSGA-II is widely applied as

[*]Corresponding author:Faculty of Engineering, Kharazmi University, Tehran, Iran
[†]E-mail address:shahruzi@khu.ac.ir

a popular algorithm due to its parameter-less sharing evaluation, elitist approach and lower computational cost with respect to other evolutionary methods. Another category of multi-objective optimization, however, utilizes vector-sum search engines which provide considerably more efficient algorithms. Hu and Eberhart proposed a Dynamic Neighborhood Particle Swarm Optimization using a similar approach to lexicographic methods; in which only one objective is optimized at a time [6]. Ray and Liew [7] utilized Pareto dominance and crowding control in Particle Swarm Optimization to develop a multi-objective search with desired diversity. Coello et al. [8] offered a Multi-Objective Particle Swarm Optimization with an additional repository memory, an extra exploratory operator and a constraint handling scheme. They reported higher efficiency of MOPSO in deriving competitive results to three other MOO algorithms [8-10] especially in convergence to true PF of their test functions.

All these methods require true judgment for their parameter thresholds. Parameter tuning in crisp type may bring about undesired consequences or require several time-consuming trial runs for each specific problem. An alternate approach is to utilize expert recommendations or previous experiences via a more comprehensive logic which can simulate some aspects of human thinking in automatic algorithms.

Since 1965, when Prof. Zadeh introduced *Fuzzy Logic*, FL [11], it has undergone considerable developments in several engineering fields [12-14].One interesting point that FL introduces is that imprecise definitions within the inference core may result in better results when de-fuzzified for practical implementation.The matter concerns areas with lack of data or difficulty in crisp inference.

One such field is parameter tuning of optimization algorithms. According to No-Free-Lunch theorem [15], no specific set of parameters are expected to be the most suitable for all kinds of problems. In another word, exact parameter tuning relies on having the complete problem-specific data. However, it is not the case in practice as the design space areas are not brightened before starting the search via an optimization algorithm. Perhaps, the search space will not be perfectly brightened even after that, as we define an efficient algorithm the one to find the optimum with minimal sampling effort. Nevertheless, it should be noted that some algorithms with few parameters have already been developed in recent years that require less tuning effort [16-21].

The issue is concerned here by extending the definition of crisp values for the algorithm parameters to fuzzy membership functions. This approach is further implemented on the framework of multi-objective particle swarm optimization. Performance of the proposed method is evaluated treating a number of literature benchmarks including test functions and structural sizing examples.

2. APPLIED FUZZY CONCEPTS AND OPERATORS

Theory of *Fuzzy Sets* and *Fuzzy Logic* can be found in several references [22-23]. Here, a review of some basic concepts is briefed as required in later sections.

2.1 Fuzzy set

A *fuzzy set*, L, is introduced by a membership function; $\mu_L(\mathrm{y})$, that indicates the degree of belief that how much its *Linguistic Variable*, *y*, belongs to a *linguistic label*, *L*. This label is indeed the name of that fuzzy set and is alternatively called *Linguistic Value*. The domain, on which membership has a non-zero value, is called *support* of that fuzzy set. A continuous fuzzy set has a continuous support. The membership function varies between 0 and 1 indicated by:

$$0 \le \mu_L(\mathrm{y}) \le 1 \tag{1}$$

2.2 Normal fuzzy set

A fuzzy set that includes at least one point with full membership grade; that is:

$$\exists y \in Support(L): \ \mu_L(\mathrm{y}) = 1 \tag{2}$$

2.3 Fuzzy variable

A *fuzzy variable* includes:a linguistic variable, its domain of variation called *universe of discourse*, a number of linguistic *Labels* and their corresponding *membership functions*.

2.4Fuzzy partition

Consider the case that every point on the universe of discourse belongs to at least the support of one fuzzy label. Such a fuzzy variable introduces a *fuzzy partition* over the corresponding universe of discourse.

2.5 Fuzzy singleton

A normal fuzzy set including only one point y_c in its support. It indeed reveals an alternate definition of a*crisp value*; i.e. the case that the belief for membership of y_c to the set is 1 only at y_c and 0 at the other points.

2.6Complement/ concentration/dilution operators

For each given fuzzy set, L , its *simple fuzzy complement* is defined by another set $\neg L$ (the same as $c(L)$) on the same support set so that

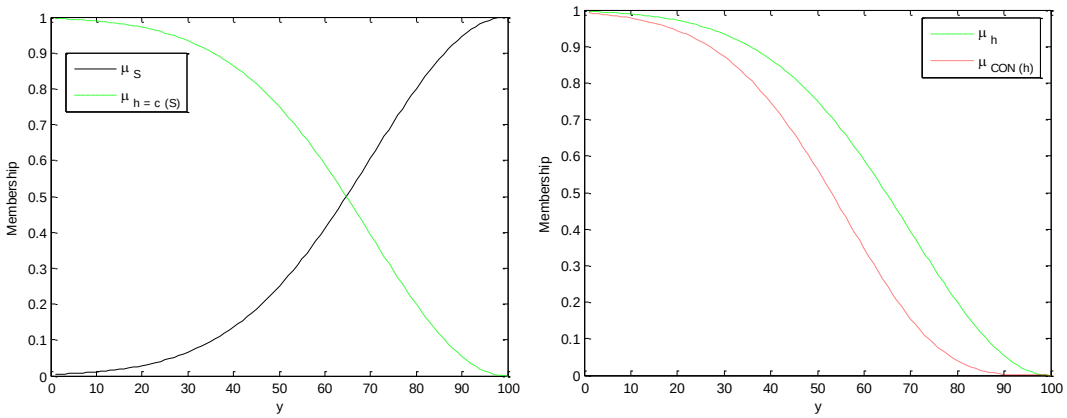
$$\forall y \in Support(L): \ \mu_{\neg L}(\mathrm{y}) = 1 - \mu_L(\mathrm{y}) \tag{3}$$

A given linguistic value, L , may be *concentrated* or *diluted* if desired. A simple definition for such operators can be revealed as follows.

$$\forall y \in Support(L): \ \mu_{CON(L)}(\mathrm{y}) = (\mu_L(\mathrm{y}))^n, \quad n > 1 \tag{4}$$

$$\forall y \in Support(L): \ \mu_{DIL(L)}(\mathrm{y}) = (\mu_L(\mathrm{y}))^n, \quad 0 < n < 1 \tag{5}$$

In another word, $\mu_{CON(L)}(\mathrm{y}) < \mu_L(\mathrm{y})$ for most parts of the support; that means stricter membership regulation is exerted by concentrating a linguistic value (Fig.1) and vice versa for dilution operator.



(a) (b)

Figure 1. Sample fuzzy set and its (a) fuzzy complement, (b) fuzzy concentrated set

3. MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION

A wide spectrum of meta-heuristic methods, are developed in recent years which apply vector-sum partitioning of the design space. *Particle Swarm Optimization*, PSO is perhaps the pioneer work in this class and a very popular optimization tool for engineering applications [24-26].

Any particle in PSO denotes a design vector, X; which is evaluated at any location in the design space to determine its corresponding fitness score. Movement of each Particle, X, from its current location at iteration, k, toward a new position in the next iteration obeys from the following rules:

$$\vec{X}_i^{k+1} = \vec{X}_i^{k} + \vec{V}_i^{k+1} \tag{6}$$

$$\vec{V}_i^{k+1} = Q_I^k \vec{V}_i^{k} + Q_C^k(\vec{B}_i^{k} - \vec{X}_i^{k}) + Q_S^k(\vec{G}^{k} - \vec{X}_i^{k}) \tag{7}$$

in which $\vec{B}_i^k$ denotes the local-best experience of the i^{th} particle up to the iteration, k, and $\vec{G}^k$ is the global best of all particles. $\vec{V}_i^{k+1}$. As a result the new velocity vector of any i^{th} particle is obtained by summation of three scaled movements at the following directions:

– Inertial direction: moving parallel to the last velocity vector of the particle, $\vec{V}_i^k$ scaled by the factor Q_I^k

– Cognitive direction: moving from the current position $\vec{X}_i^k$ toward the previous best experience of each particle itself; $\vec{B}_i^k$. The corresponding scale factor is denoted by Q_C^k.
– Social direction: moving from the current position $\vec{X}_i^k$ toward the global best position; $\vec{G}^k$ scaled by the factor Q_C^k

In the traditional PSO, the cognitive and social factors are given by:

$$\begin{aligned} Q_C^k &= rand \times C_C \\ Q_S^k &= rand \times C_S \end{aligned} \tag{8}$$

The function *rand* uniformly generates random values in range [0, 1]. The PSO parameters: C_S and C_C controls the relative importance of global or local best experience of particles in the total movement vector. It is while the inertial term has an exploitative effect. Hence, it is offered by experts to be decreased as the search progresses to the last iteration, N_{Iter} [27].

In the traditional method of parameter tuning, crisp values should be assigned to such factors. One may ask how precise are these crisp values for optimization? The answer, however, relies on the number of trial runs to tune them and also the explorative effect of the *rand* function. Besides, dissimilarities of the search space from case to case, makes it difficult to fix a parameter set that is best suited for all possible cases.

Existing vagueness in answering the aforementioned question leads us to switch into application of fuzzy variables, as the alternative. Suppose the following facts are accepted regarding the general PSO parameters Q_I^k, Q_C^k, Q_S^k:

1. Fact-1:None of the inertial, cognitive and social terms is limited to an especial iteration
2. Fact-2: Belief to the truth of the global best at the last iteration is the highest among the entire search process.
3. Fact-3: As the belief to truth of $\vec{G}^k$ is strengthened via the search progress, necessity of diversification about it to find better solutions is weakened.
4. Fact-4:The inertial factor is desired to have its maximum at early iterations
5. Fact-5:Importance of the inertial factor is desired to vanish faster than increasing the social term
6. Fact-6: Importance of the local best (cognitive term) in the search direction is lower than the global best (social term) near the end of the optimization process. In addition, the belief to truth of $\vec{B}_i^k$ is not considerable at early iterations.

The first fact, indicate that crisp definition of PSO parameters can be extended to fuzzy partition over the iteration as a linguistic variable. It is evident that the universe of discourse is defined by integers from 1 to N_{Iter}.

A simple yet continuous expansion of a crisp value at m to a fuzzy set can be defined by the following Gaussian-like distribution function:

$$\mu(k) = N(k, y_c, \sigma) = \beta \, \mathrm{e}^{-\frac{1}{2}\left(\frac{k-y_c}{\sigma}\right)^2} \tag{9}$$

in which y_c is the median and σ denotes the standard deviation. β is conceptually 1, however, can be taken 2 for efficiency reasons. Lowering the standard deviation acts like a concentration operator for this function. The maximum of N(.) occurs at the median/mean i.e. $N(y_c, y_c, \sigma) = 1$.Another interesting feature of this fuzzy set is that its support covers all values in the universe of discourse. It deserves Fact-1 requirement.

Let's use the following fuzzy set for the PSO social parameter Q_S^k. It introduces an increasing function which takes its maximum at the last iteration (Fact-2) and non-zero values at the others (Fact-1).

$$\mu_S(k) = N(k, N_{Iter}, \sigma) \tag{10}$$

The membership function for the inertial term is at first considered the complement of $\mu_S(k)$ due to Fact-3 and Fact-4.A concentration operator can also act on it to satisfy Fact-5. The resulting function is given by:

$$\mu_I(k) = (1 - \mu_S(k))^2 \tag{11}$$

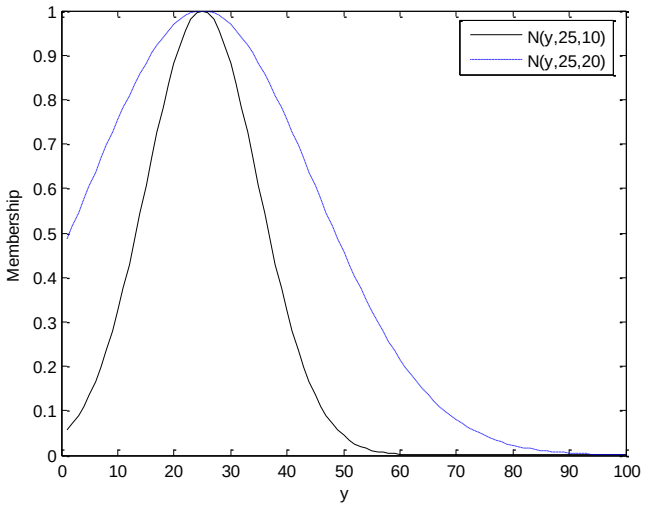


Figure 2. Effect of σ variation on the shape of N(.) as membership function

Similarly, fuzzy set of the cognitive term is characterized by

$$\mu_C(k) = N(k, y_C, \sigma) \tag{12}$$

In order to satisfy Fact-6, the median y_C for the cognitive membership function falls between the 1st and the end iterations by.

$$y_C = rand \times N_{Iter} \tag{13}$$

The coefficient σ is also taken a random portion of $2N_{Iter}$ (Fig.2). Finally, the PSO parameters are tuned by the aforementioned fuzzy partition so that:

$$< Q_I^k, Q_C^k, Q_S^k > =< \mu_I(k), \mu_C(k), \mu_S(k) > \tag{14}$$

Fig.3 demonstrates a sample fuzzy partition over the iteration counter k.

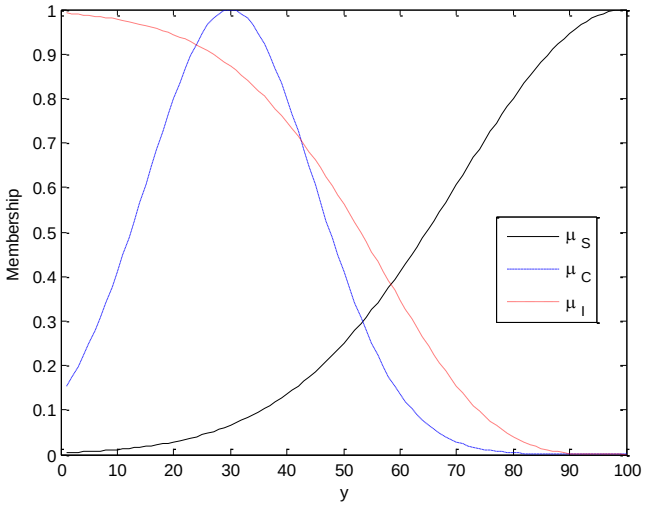


Figure 3. Sample fuzzy partition for the inertial, cognitive and social memberships in FPSO

PSO analogy can be applied into a multi-objective framework keeping the same structure of EQ.(6) and (7) but with some changes in evaluating its terms. First, Pareto-dominance and fitness sharing controller are used to adapt new definitions for the local/global best terms in Eq.(7).Second, an auxiliary archive called *repository* is employed to save desired number of non-dominated solutions found up to the current iteration. Third, a mutation operator is added to avoid the algorithm from premature convergence to a false PF (that is equivalent to a local optimum in global optimization) [8]. In this regard, we employ the above actions with implementation of our Fuzzy tuning scheme. The new algorithm called *Multi-Objective Fuzzy Particle Swarm Optimization;* MOFPSO is, thus, introduced via the following steps:

1. ***Initialization***

1.1. Generate the population of particles; POP by randomly locating them within their limits:

$$\vec{X}_i^0 = \vec{X}_{LB} + rand \otimes (\vec{X}_{UB} - \vec{X}_{LB}) \tag{15}$$

Where $\otimes$ indicate a component-wise product. $\vec{X}_{LB}$ and $\vec{X}_{UB}$ denote *lower* and *upper bounds* on design variables, respectively. Every design vector had N_{Var} variables while the

number of particles is N_{POP}. Set limit of *Repository size*; $N_{REP} = N_{Pop}$ and the number of iterations; N_{Iter}

1.2. Evaluate cost function vector for each particle

1.3. Initiate the Repository: Extract non-dominated solutions out of the current population and store them in Repository Archive; REP.

1.4. *Construct the Grid*: take every cost function as a coordinate and divide the line between already-found minimum and maximum of that function by N_{Grid} parts. Intersection of such grid parts forms a number of hyper-cubes; into which the current Pareto front is subdivided

1.5. Set each particle's velocity to zero.

$$\vec{V}_i^0 = \vec{0} \tag{16}$$

1.6. Initialize local best positions of the particles by their current positions

$$\vec{B}_i^0 = \vec{X}_i \tag{17}$$

2. Loop until the loop counter k ***reaches the prescribed value:*** N_{Iter}

2.1. Determine global best particle $\vec{G}^k$ out of REP.

2.1.1. Construct hyper-cubes using the current grid

2.1.2. Calculate any hyper-cube's fitness by the following relation when n_h represents the number of Pareto points within that hyper-cube

$$F_h = \frac{1}{n_h} \tag{18}$$

2.1.3. Select the fittest hyper-cube by a roulette-wheel procedure

2.1.4. Choose one of the Pareto solutions in the selected hyper-cube as the current global best: $\vec{G}^k$

2.2. Update Fuzzy sets $\mu_I(k), \mu_C(k), \mu_S(k)$ by Eq.(10~12) and the mutation threshold by:

$$P_m = \mu_I(k)\mu_C(k) \tag{19}$$

2.3. For every i^{th} particle do:

2.3.1. Calculate the particle velocity using Eq.(7)

2.3.2. Update the position of that particle by Eq.(8)

2.3.3. If $rand < P_m$ then apply mutation on the particle,

2.3.4. Correct the particle's position so that it falls within its prescribed bounds

2.3.5. Evaluate cost function vector for the particle and its mutated variant

2.3.5.1. Replace the particle with its mutated variant if the particle is dominated by such a newcomer

2.3.5.2. else if the particle does not dominate its mutated variant replace them with 50% probability

2.4. Apply Pareto dominance to update local best positions:

2.4.1. If the current particle $\vec{X}_i^k$ dominates its previous-best position $\vec{B}_i^{(k-1)}$ update its local best by $\vec{B}_i^k = \vec{X}_i^k$

2.4.2. else if the current particle $\vec{X}_i^k$ is dominated by its previous-best position $\vec{B}_i^{(k-1)}$ then $\vec{B}_i^k = \vec{B}_i^{(k-1)}$

2.4.3. otherwise randomly select one of the $\vec{X}_i^k$ and $\vec{B}_i^{(k-1)}$ as $\vec{B}_i^k$

2.5. ***Update the Repository and Grid***

2.5.1. Extract non-dominated solutions out of the current population and those in the current REP

2.5.2. If the number of these non-dominated solutions are less than N_{REP} take them as the updated REP;

2.5.3. otherwise eliminate extra non-dominated solutions from REP until the remained ones form an updated REP with N_{REP} members. In this procedure, give priority of elimination to the ones that are located in more crowded hyper-cubes.

2.5.4. Construct new grid using the updated REP

2.6. *Increment the loop counter*

3. Announce the REP members as the final Pareto Front

Flowchart of the proposed MOFPSO is given in Fig.4. The external archive REP for non-dominated solutions; has as an elitism saving role in multi-objective frame work and also acts a guide for particle movements in the corresponding MOFPSO.

Meanwhile, the idea behind applying hyper-cubes is to find well-distributed Pareto fronts by constant grid divisions. This adaptive grid calculation has been reported to have less complexity than niching process in related literature [8, 27].

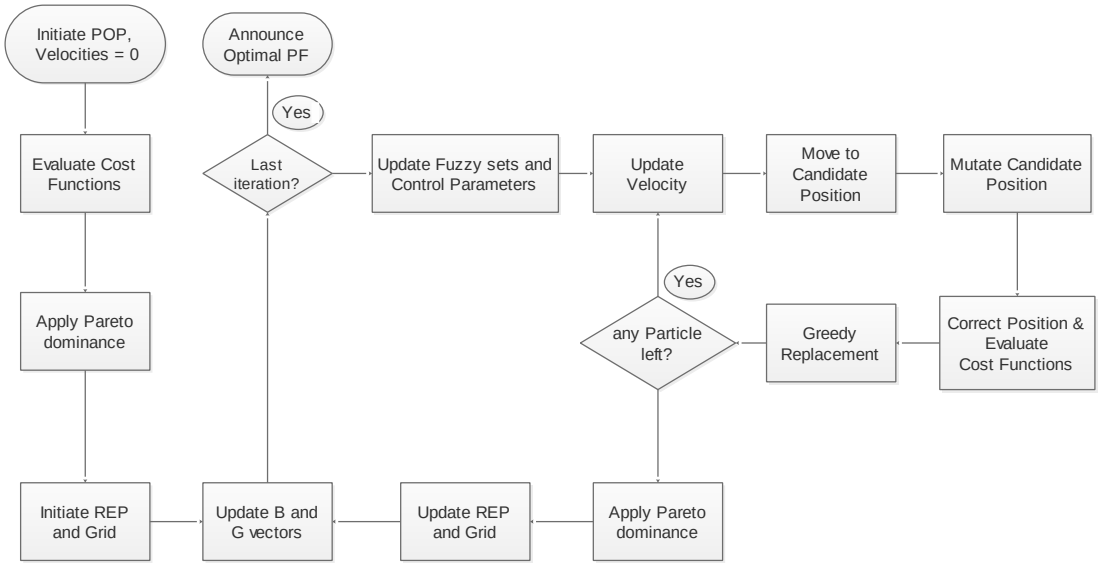


Figure 4. Flowchart of the proposed MOFPSO

4. PERFORMANCE METRICS

Desirability of the Pareto front resulted by MOO algorithm requires some different measures from those of single-objective optimization.For example, the global optimum is no more a single point; but is denoted by the points (or an equation) coordinating true PF. The equation covering such a set is available for some MOO test functions; however, it is not the case for most of the engineering problems. Therefore, it is common to use test functions for evaluation purposes. In such a case, a *distancein the functions view* between the final non-dominated solutions of a MOO algorithm from true Pareto front will be a measure of its quality of convergence.

As another feature, a desired MOO result should reveal a good *spread* of points along the Pareto front; i.e. as uniform as possible. MOO algorithm should also be powerful in capturing corners of true Pareto front as well as its interior regions. In another word, maximum *extension* of PF is desired to be achieved. The employed measures in this study are listed below.

4.1 Generationaldistance metric

In order to asses the distance between non-dominated set and true Pareto front, we apply a *Generational Distance*, GD; based on that introduced by Van Veldhuizen and Lamont [29] as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} e_i^2}}{n} \tag{20}$$

where n is number of solutions in Pareto set and e_i is the smallest Euclidean distance between each position and true optimal set.

The minimal GD is zero indicating that all resulted Pareto points by the algorithm are on true Pareto front; i.e. the MOO global optimum.

4.2 Spacing metric

The spacing metric introduced by Schott [30] deals with the spread (distribution quality) of the Pareto front.

$$SP = \sqrt{\frac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n-1}} \tag{21}$$

whereas for any i^{th} point on the Pareto front we have:

$$d_i = \min_j \sum_{s}^{N_f} \left| f_s^j(\vec{X}) - f_s^i(\vec{X}) \right| \tag{22}$$

N_f stands for the number of the cost functions and $\bar{d}$ is the average over all d_i. The lower SP corresponds to better spread of Pareto front [30].

4.3 Extent metric

Zitzler et al. [31] introduced this metric on the resulted Pareto front. It is defined as follows:

$$EX = \sqrt{\sum_{i=1}^{n} D_i} \tag{23}$$

$$D_i = \max_j \sum_{s}^{N_f} \left| f_s^j(\vec{X}) - f_s^i(\vec{X}) \right| \tag{24}$$

For a good MOO result EX is desired to be as large as possible.

4.4 CPU Time

As the metric for computational efficiency; the elapsed *CPU Time*, CT (in seconds) is evaluated by running all the algorithms on the same platform.

5. NUMERICAL INVESTIGATION

Performance of the proposed MOFPSO is evaluated here in comparison with the well-known NSGA-II on a variety of MOO test functions and sizing problems. Every example is

solved by both the algorithms using 50 independent runs with $N_{Pop}=100$ and $N_{Iter}=200$. Specific parameters of NSGA-II are tuned as $n_{Crossover}=1.00$, $n_{Mutation}=0.40$ and $P_{Mu}=0.02$ while $N_{Grid}=7$ for MOFSPO.

A number of problems are then treated by the aforementioned MOO algorithms. All the algorithms are programmed in MATLAB and run on a 64^{bit} platform with CORE^{TM}i3-3120M 2.5^{GHz} CPU and 4^{GB} RAM. It is worth mentioning that in every run the random initial population is identically used by the algorithms for true comparison.

The first 3 MOO test functions are given by Deb [32] for $N_{var}=30$ and $x_i \in [0,1]$ as follows.

Test Problem 1: This test function has a convex Pareto optimal front. TP1 is defined by:

$$\textit{Minimize} \qquad \vec{f}(\vec{X})=\begin{cases} f_1(\vec{X})=x_1 \\ f_2(\vec{X})=p\left(1-\sqrt{\dfrac{x_1}{p}}\right) \end{cases} \tag{25}$$

where:

$$p=1+9\sum_{i=2}^{N_{var}}\frac{x_i}{N_{var}-1} \tag{26}$$

Test Problem 2: True Pareto front of this example (TP2) consists of several non-continuous convex parts. It is formulated as:

$$\textit{Minimize} \qquad \vec{f}(\vec{X})=\begin{cases} f_1(\vec{X})=x_1 \\ f_2(\vec{X})=p\left(1-\sqrt{\dfrac{x_1}{p}}-\dfrac{x_1}{p}\sin(10\pi x_1)\right) \end{cases} \tag{27}$$

$$p=1+\frac{9}{N_{var}-1}\sum_{i=2}^{N_{var}}x_i \tag{28}$$

Test Problem 3: This function has a non-convex Pareto set as defined by:

$$\textit{Minimize} \qquad \vec{f}(\vec{X})=\begin{cases} f_1(\vec{X})=1-\exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(\vec{X})=q\left[1-\left(\dfrac{x_1}{q}\right)^2\right] \end{cases} \tag{29}$$

where:

$$q = 1 + 9\left(\sum_{i=2}^{N_{\text{var}}} \frac{x_i}{N_{\text{var}} - 1}\right)^{0.25} \tag{30}$$

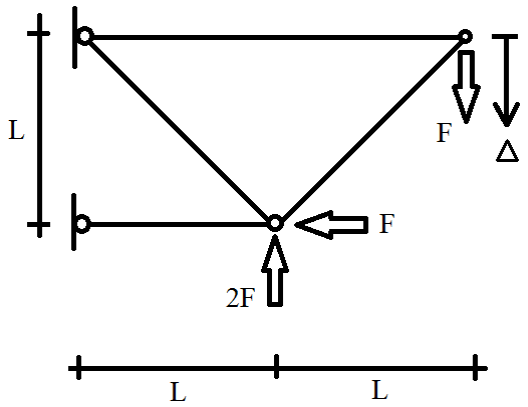


Figure 5. Four-bar truss of the test problem-4

Test Problem 4: Sizing of this 4-bar truss is concerned with 4 stress constraints as introduced in [33]. The constraints are however re-formed to be assessed like variable limits.

The problem has two objectives; i.e. total structural volume and the upper corner node's displacement Δ. Member cross sectional areas form the design variables. Formulation of this problem is given as.

$$\textit{Minimize} \qquad \vec{f}(\vec{X}) = \begin{cases} f_1(\vec{X}) = L\left(2x_1 + \sqrt{2}x_2 + \sqrt{x_3} + x_4\right) \\ f_2(\vec{X}) = \dfrac{F.L}{E}\left(\dfrac{2}{x_1} + \dfrac{2\sqrt{2}}{x_2} - \dfrac{2\sqrt{2}}{x_3} + \dfrac{2}{x_4}\right) \end{cases}$$

$$\textit{Subject to}: \tag{31}$$

$$\frac{F}{\sigma} \le x_1 \le 3\frac{F}{\sigma}$$

$$\sqrt{2}\frac{F}{\sigma} \le x_2 \le 3\frac{F}{\sigma}$$

$$\sqrt{2}\frac{F}{\sigma} \le x_3 \le 3\frac{F}{\sigma}$$

$$\frac{F}{\sigma} \le x_4 \le 3\frac{F}{\sigma}$$

where $F = 10kN$, $\sigma = 10\,{kN}/{cm^2}$, $E = 2\times10^5\,{kN}/{cm^2}$, $L = 200\,cm$.

Test Problem 5: It is a two-bar truss sizing problem, studied by a number of researchers [34, 35]. The truss model is demonstrated in Fig.6. It has to carry the vertical load without elastic failure. Consequently, one of the problem's objectives is to minimizing stress in each bar and another one is to minimize total volume of the truss. Design variables x_1, x_2, x_3 indicate section areas of the bar elements and vertical distance between the node C and the supports, respectively.

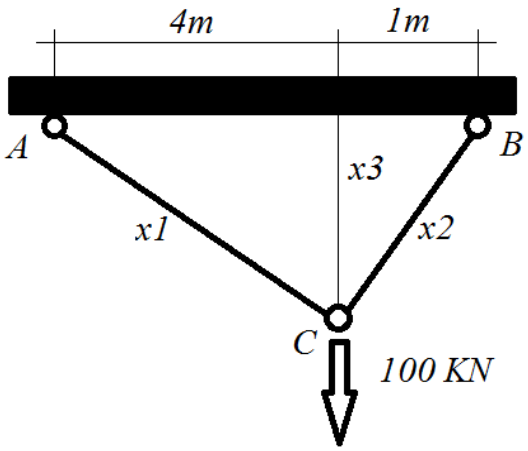


Figure 6. Two-bar truss of the test problem-5

The mathematical description of problem is expressed as follows:

$$
\begin{aligned}
&\textit{Minimize} \qquad \vec{f}(\vec{X}) = \begin{cases} f_1(\vec{X}) = x_1\sqrt{16+x_3^{\,2}} + x_2\sqrt{1+x_3^{\,2}} \\ f_2(\vec{X}) = \max(\sigma_{AC}, \sigma_{BC}) \end{cases} \\
&\textit{Subject to}: \\
&0 \le x_1 \\
&0 \le x_2 \\
&1 \le x_3 \le 3 \\
&\max(\sigma_{AC}, \sigma_{BC}) \le 10^5
\end{aligned}
\tag{32}
$$

$$
\textit{where } \sigma_{AC} = \frac{20\sqrt{16+x_3^{\,2}}}{x_1 x_3},\ \sigma_{BC} = \frac{80\sqrt{1+x_3^{\,2}}}{x_2 x_3}
$$

Test Problem 6: TP6 is a frame sizing design problem. The three-story moment frame is shown in Fig.7 for which $L = 6m$ and $H = 4m$. Floor diaphragms are rigid and axial deformations are neglected. Thus, the system has only one degree of freedom at each story. Total live and dead load at each floor is given 56 KN/m that does not include

the columns weight. Elastic module and material density of the employed steel are taken as $200\times10^{6}\ kN/m^{2}$ and $7800\ kg/m^{3}$, respectively.

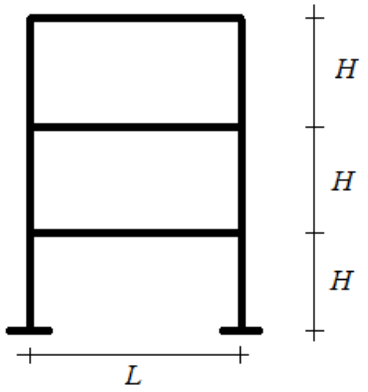


Figure 7. Moment frame of the test problem-6

Design variables can take column section ID's. Each ID may be associated an integer as in the available section list of Table 1. It is, therefore, a discrete problem with two conflicting objectives: the structural weight and its fundamental period.

Table 1: The employed section list for the frame sizing problem

ID	Section	Cross area (cm^2)	Moment inertia (cm^4)
1	IPE 8	7.64	80.1
2	IPE 10	10.3	171
3	IPE 12	13.2	318
4	IPE 14	16.4	541
5	IPE 16	20.1	869
6	IPE 18	23.9	1320
7	IPE 20	28.5	1940
8	IPE 22	33.4	2770
9	IPE 24	39.1	3890
10	IPE 27	45.9	5790
11	IPE 30	53.8	8360
12	IPE 33	62.6	11770
13	IPE 36	72.7	16270
14	IPE 40	84.5	23130
15	IPE 45	98.8	33740
16	IPE 50	116	48200
17	IPE 55	134	67120
18	IPE 60	156	92080

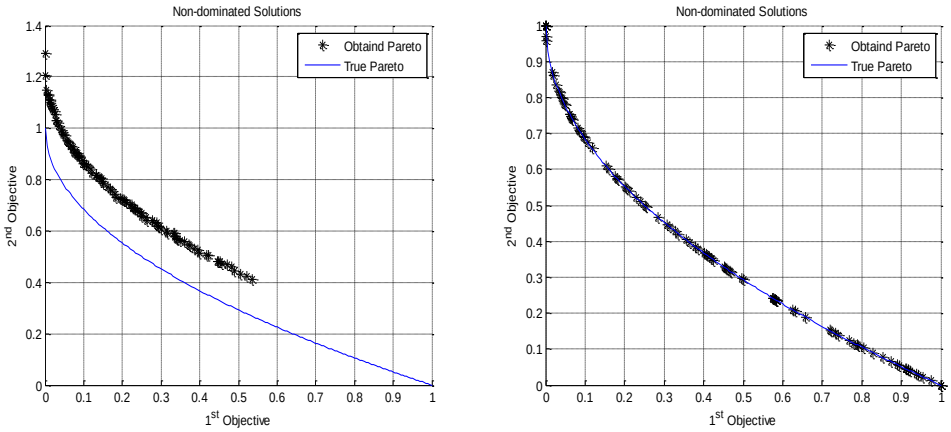
In order to prevent formation of soft story, an extra constraint is also applied as:

$$K_i \le 0.7K_{i+1} \tag{33}$$

Fig. 8 shows that despite NSGA-II, the proposed MOFPSO has been successful in capturing true Pareto front of TP1 in the same 200 iterations.

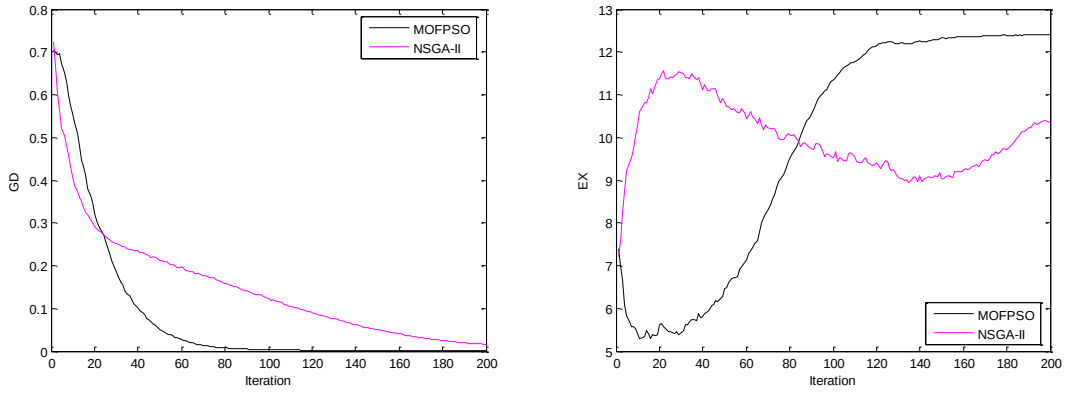
Trace of GD in Fig. 9 declares that MOFPSO has much faster convergence with respect to NSGA-II. This figure shows completely different trend of EX variation. It has converged to a high maximum after an early falling downfor MOFPSO. Note that higher EX is desired for a proper MOO solution.

Table 2 summarizes metric statistics for TP1 to TP4 for which true PF curve is given in literature. It confirms such GD superiority not only in average but also in the best and worst run results. However, NSGA-II has obtained lower SP measure than MOFPSO. Regarding the 3rd metric; i.e. EX, MOFPSO has been superior in capturing the extents of true Pareto front, as can also be realized from Fig. 8.



(a) (b)

Figure 8.Resulted Pareto front by (a) NSGA-II and (b) MOFPSO for TP1



(a) (b)

Figure 9. Comparison of MOFPSO vs. NSGA-II in (a) GD and (b) EX traces for TP1

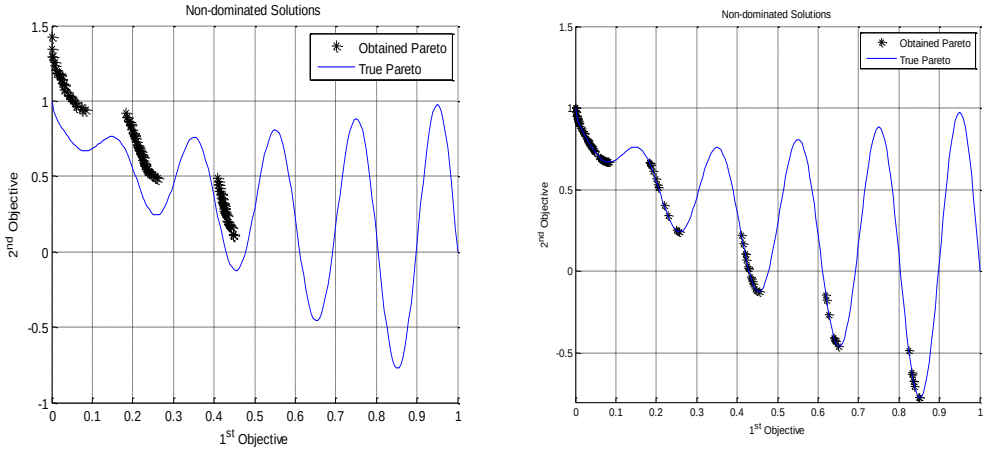
Table 2: Statistical results of MOO test problems with known PF curve

Test Problem	Metric		NSGA-II	MOFPSO
TP1	GD	Mean±SD	0.0159±0.0021	0.0004±0.0003
		Best	0.0125	0.0001
		Worst	0.0213	0.0014
	SP	Mean±SD	0.0088±0.0036	0.0100±0.0012
		Best	0.0045	0.0081
		Worst	0.0210	0.0133
	EX	Mean±SD	10.3471±0.9530	12.4095±0.0432
		Best	12.5920	12.4933
		Worst	8.6650	12.2839
	CT	Mean±SD	336.1350±1.3504	55.6203±3.6171
		Best	334.0469	49.2813
		Worst	342.4688	62.3906
TP2	GD	Mean±SD	0.0189±0.0041	0.0028±0.0048
		Best	0.0110	0.0001
		Worst	0.0269	0.0240
	SP	Mean±SD	0.0128±0.0061	0.0192±0.0096
		Best	0.0077	0.0079
		Worst	0.0381	0.0648
	EX	Mean±SD	12.0251±1.2639	11.5381±1.7828
		Best	14.4059	14.7162
		Worst	9.3426	6.9458
	CT	Mean±SD	337.0581±2.1457	26.0600±6.3048
		Best	332.5313	19.8281
		Worst	346.8125	52.56
TP3	GD	Mean±SD	0.0130±0.0129	0.0473±0.0827
		Best	0.000238	0.000209
		Worst	0.0471	0.5559
	SP	Mean±SD	0.0257±0.0380	0.0598±0.0907
		Best	0.0053	0.0062
		Worst	0.1663	0.4539
	EX	Mean±SD	13.0157±2.2899	16.1796±4.8268
		Best	20.14	28.3973
		Worst	7.3141	8.4776
	CT	Mean±SD	352.0560±5.5322	51.8822±13.6215
		Best	344.2800	20.6563
		Worst	369.2100	72.0938
TP4	GD	Mean±SD	0.0134±0.00068	0.0134±0.000714
		Best	0.0118	0.0119
		Worst	0.0149	0.0148
	SP	Mean±SD	7.0470±0.6247	9.4485±1.0506
		Best	5.6161	6.9412
		Worst	8.4350	12.0457
	EX	Mean±SD	357.4608±1.6468	360.9776±0.9749
		Best	360.5267	363.1928
		Worst	354.8602	359.0310
	CT	Mean±SD	334.3119±4.7198	78.3381±3.4488
		Best	328.4219	71.9219
		Worst	347.8281	85.0000

It is interesting that MOFPSO has consumed considerably lower computational effort to obtain such results; i.e. 55.6^{s} (in average of 50 runs) which is almost on-sixth of 336.1^{s} by NSGA-II.

Treating the second test problem (TP2), it is observed that MOFPSO has rapidly converged to the curve of true Pareto front while NSGA-II has not yet captured it. Fig.10 shows that despite NSGA-II, the MOFPSO result has been extended to all valleys of non-convex Pareto curve. CT comparison in Table 2 reveals that in average MOFPSO has been about 13 times faster than NSGA-II in this example. However, regarding other metrics; i.e. SP and EX they have shown closer performance.

In optimization of TP3, GD in the best runs of MOFPSO is again better than NSGA-II, however, it is not the case for the mean GD. Like the other examples, the spread metric SP of NSGA-II has been lower than MOFPSO; but such superiority is reversed for EX regarding Table 2. Fig.11 shows that in this example, both the methods have revealed their final non-dominated solutions on the curve of true PF.

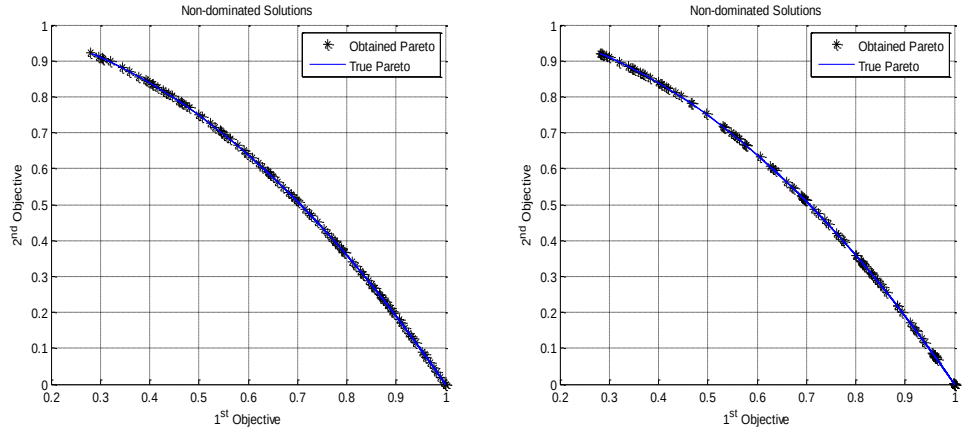


(a) (b)

Figure 10.Resulted Pareto front by (a) NSGA-II and (b) MOFPSO for TP2

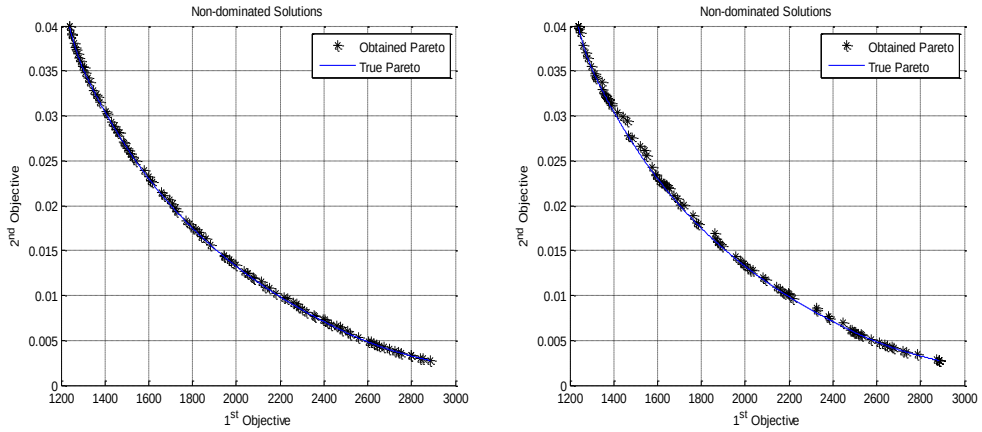
According to Fig. 12, both NSGA-II and MOFPSO have almost captured the curve of true PF. The matter is confirmed by reported GD results in Table 2. In TP4, the best method to maintain extension of PF is MOFPSO; meanwhile NSGA-II reveals the best SP results.

It is also notable that the time complexity of MOFPSO (with 78^{s} mean-CT) is still lower than NSGA-II (taking 334seconds in average). These values are updated to 71.9^{s} vs. $328.4^{s}$for the best run of the algorithms, respectively.



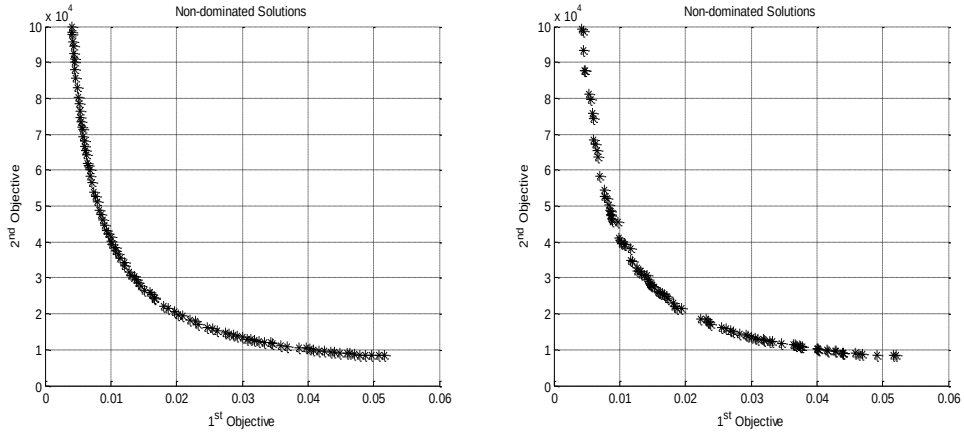
(a) (b)

Figure 11.Resulted Pareto front by (a) NSGA-II and (b) MOFPSO for TP3



(a) (b)

Figure 12.Resulted Pareto front by (a) NSGA-II and (b) MOFPSO for 4bar truss design (TP4)



(a) (b)

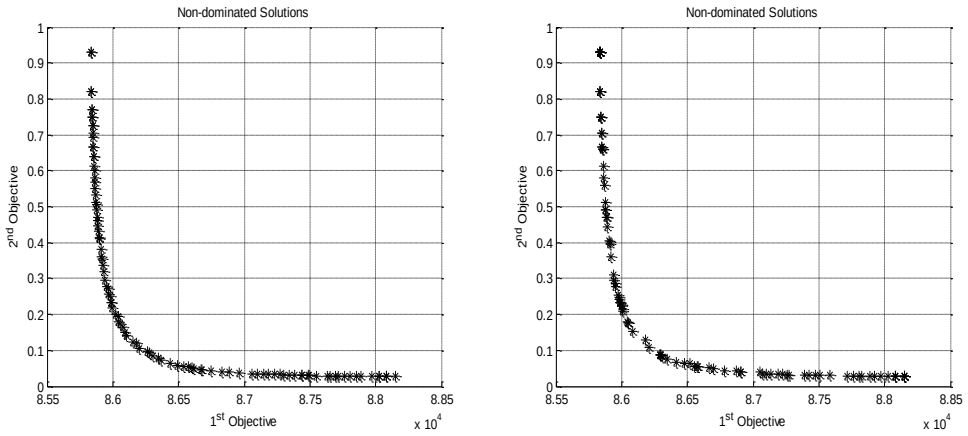
Figure 13.Resulted Pareto front by (a) NSGA-II and (b) MOFPSO for the truss sizing at TP5

Table 3: Statistics of MOO results for test problems TP5 and TP6

Test Problem	Metric		NSGA-II	MOFPSO
TP5	SP	Mean±SD	571.6212±36.2179	829.1462±211.9612
		Best	481.4095	425.4678
		Worst	647.9019	1359.8
	EX	Mean±SD	2727.8000±19.2030	2655.3000±96.0512
		Best	2753.9000	2776.8000
		Worst	2635.8000	2338.6000
	CT	Mean±SD	315.3541±2.7496	57.0097±5.6882
		Best	313.1563	39.5781
		Worst	324.4375	69.9531
TP6	SP	Mean±SD	13.6906±2.2778	18.2808±2.4223
		Best	9.8298	12.6576
		Worst	18.3005	23.6849
	EX	Mean±SD	395.3538±26.2187	432.4208±23.1425
		Best	445.7590	448.8558
		Worst	351.8258	351.7102
	CT	Mean±SD	350.2038±7.5138	72.3719±2.3545
		Best	339.3594	68.8750
		Worst	372.1563	79.5625

TP5 is a sizing problem for which true PF is not reported. Comparing Fig. 13a with Fig. 13b shows good agreement in capturing PF by the treated methods. This is also true for TP6 according to Fig. 14. However, some differences in spread of solutions along PF can be noticed in each case; that is numerically declared in Table 3.

For TP5 and TP6, MOFPSO has resulted in PF with better EX than NSGA-II in the best run; however, the average EX of TP5 is slightly lower for MOFPSO. In the treated cases, NSGA-II has revealed smaller SP but MOFPSO has taken much better CPU time applying the same N_{Pop} and N_{Iter}.



(a) (b)

Figure 14.Resulted Pareto front by (a) NSGA-II and (b) MOFPSO for the frame design (TP6)

6. CONCLUSION

The present work concerned a fuzzy approach to reduce parameter tuning effort in a multi-objective particle swarm optimization. In this regard, the algorithm parameters are fuzzified taking into account some prescribed suggestions about proper iteration-wise variation of these factors.

Gaussian distribution is utilized to generate fuzzy partition of the corresponding factors. Consequently, the cognitive and social coefficients are assigned a type of normal fuzzy sets. The other factors are constructed by fuzzy complement and concentration operators on the aforementioned membership functions. Such fuzzy sets depend on some random seeds, however, their maximal height are pre-determined. In fact, they represent especial types of control factors for the proposed MOFPSO.

Statistical results of optimization in a number of test functions and also sizing design examples showed that MOFPSO is capable of capturing true Pareto front. In another word, it can provide sufficient diversity by its randomized terms to escape from local optima. In most treated problems MOFPSO resulted in more extended Pareto front with respect to NSGA-II, however, with evenly distributed points on PF.

Parameter-less structure of the proposed method is of practical interest. Although, such a feature does not allow the best tuning for maximal performance, MOFPSO was competitive to NSGA-II in our tests regarding GD and EX metrics. In addition, computational time efficiency of MOFPSO is beneficial for solving engineering problems.

REFERENCES

1. Schaffer JD. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Proceedings of the 1st International Conference Genetic Algorithms*, Pittsburgh, PA, USA, July 1985.
2. Fonseca CM, Fleming PJ. Genetic Algorithms for Multi-objective optimization: Formulation, Discussion, and Generalization, *Proceedings of the 5th International Conference Genetic Algorithm*, San Mateo, CA: Morgan Kaufmann, July 1993; pp. 416-423.
3. Horn J, Nafpliotis N, Goldberg D. A Niched Pareto Genetic Algorithm for Multi objective Optimization, *Proceedings 1st International IEEE Conference Computational Intelligence* 1994; pp. 82-87.
4. Zitzler E, Laumanns M, Thiele, L. SPEA2: Improving the Strength Pareto, *Evol Comput* 2000; 95-100.
5. Deb K, Agarwal S, Pratap, A, Meyarian T. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II, *Proceedings of the 6th International Conference Parallel Problem Solving from Nature*, PPSN VI 2000; pp. 849-858.
6. Hu X, Eberhart R. Multi-objective optimization using dynamic neighborhood particle swarm optimization, *in Proceedings of the 2002 Congress on Evolutionary Computation. (CEC'2002)*, Honolulu, HI, May 2002; **2**: pp. 1677-1681.

7. Ray T, Liew KM. A swarm metaphor for multi-objective design optimization, *Eng Opt* 2002; **34**(2): 141-53.
8. Coello Coello CA, Pulido GT, Lechuga MS. Handling Multiple Objectives with Particle Swarm Optimization, *IEEE Trans Evol Comput* 2004; **8**: 256-79.
9. Parsopoulos K, Tasoulis D, Vrahatis M. Particle swarm optimization method in multi-objective problems, *Proceedings of the 2002 ACM symposium on Applied computing* 2002; pp. 603-607.
10. Mishra BSP, Dehuri S, Cho SB. *Multi-Objective Swarm Intelligence: Theoretical Advances and Applications,* in: *Studies in Computational Intelligence*, Series Ed: Kaprzyk J. Polish Academy of Science, Springer, Warsaw, 2015; **592**: 27-73.
11. Zadeh LA. Fuzzy sets, *Info Control* 1965; **8**: 338-53.
12. Zadeh LA. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, *Fuzzy Sets and Sys* 1997; **90**: 111-27.
13. Javadian M, Bagheri-Shouraki S. UALM: Unsupervised active learning method for clustering low-dimensional data, *J Intel Fuzzy Sys* 2017; **32**(3): 2393-2411.
14. Ross TJ. *Fuzzy Logic with Engineering Applications*, 4th Ed, Wiley, UK, 2017.
15. Wolpert DH, Macready WG. No free lunch theorems for optimization, *IEEE Trans Evol Comput* 1997; **1**(1): 67-82.
16. Woldesenbet YG, Yen GG, Tessema BG. Constraint handling in multi-objective evolutionary optimization, *IEEE Trans Evol Comput* 2009; **13**(3): 514-25.
17. Cheng MY, Prayogo D. Symbiotic Organisms Search: a new meta-heuristic optimization algorithm, *Comput Struct* 2014; **139**: 98-112.
18. Kaveh A, Mahdavi VR. *Colliding Bodies Optimization, Extensions and Applications*, Springer, Switzerland, 2015.
19. Panda A, Pani S. Multi-objective colliding bodies optimization, In: Pant M, Deep K, Bansal J, Nagar A, Das K. (Eds), *Proceedings of the 5th International Conference Soft Computing for Problem Solving*, Advances in Intelligent Systems and Computing, Springer, Singapore, 2016; **436**: pp. 651-664.
20. Kaveh A. *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, 2nd edition, Springer, Switzerland, 2017.
21. Shahrouzi M, Aghabaglou M, Rafiee F. Observer-Teacher-Learner-Based Optimization: an enhanced meta-heuristic for structural sizing design, *Struct Eng Mech* 2017; **62**(5): 537-50.
22. Klir GJ, Yuan B. *Fuzzy Sets and Fuzzy Logic, Theory and Applications*, Prentice Hall, 1995.
23. Zimmermann H-J. *Fuzzy set Theory and its Applications*, 4thEd, Springer, NY, 2001.
24. Rana S, Jasola S, Kumar R. A review on particle swarm optimization algorithms and their applications to data clustering, *Artif Intel Rev* 2011; **35**(3): 211-22.
25. Kulkarni NK, Patekar S, Bhoskar T, Kulkarni O, Kakandikar GM, Nandedkar VM. Particle Swarm Optimization Applications to Mechanical Engineering- A Review, *Proceedings of the Materials Today* 2015; **2**(4–5): 2631-39.
26. Kaveh A. *Applications of Metaheuristic Optimization in Civil Engineering*, Springer, Switzerland, 2017.
27. Nickabadi A, Ebadzadeh MM, Safabakhsh R. A novel particle swarm optimization algorithm with adaptive inertia weight, *Appl Soft Comput* 2011; **11**(4): 3658-70.

28. Knowles JD, Corne DW. Approximating the non-dominated front using the Pareto archived evolution strategy, *Evol Comput* 2000; **8**:149-72.
29. Veldhuizen DA, Lamont GB. Evolutionary computation and convergence to a Pareto front, *Late Breaking Papers at the Genetic Programming Conference*, California, 1998; pp. 221-228.
30. Schott JR. Fault tolerant design using single and multi-criteria genetic algorithm optimization, Master's thesis, Department of aeronautics and astronautics, Massachusetts institute of technology, Cambridge, 1995.
31. Zitzler E, Thiele L, Deb K. Comparison of multi-objective evolutionary algorithms: empirical results, *Evol Comput* 2000; **8**: 173-95.
32. Deb K. Multi-objective genetic algorithms: Problem difficulties and construction of test problems, *Evol Comput* 1999; **7**(3): 205-30.
33. Cheng FY, Li XS. Generalized center method for multi-objective engineering optimization, *Eng Opt* 1999; **31**: 641-61.
34. Palli N, Azram S, McCluskey P, Sundararajan R. An interactive multistage ε-inequality constraint method for multiple objectives decision making, *ASME J Mech Des* 1998; **120**(4): 678-86.
35. Ray T, Tai K, Seow KC. Multi-objective design optimization by an evolutionary algorithm, *Eng Opt* 2001; **33**(4): 399-24.