Complex & Intelligent Systems (2020) 6:635–650
https://doi.org/10.1007/s40747-020-00159-y

ORIGINAL ARTICLE

Multi-objective particle swarm optimization with random immigrants

Ali Nadi Ünal[1] · Gülgün Kayakutlu[2]

Received: 25 November 2017 / Accepted: 23 May 2020 / Published online: 12 June 2020

Abstract
Complex problems of the current business world need new approaches and new computational algorithms for solution. Majority of the issues need analysis from different angles, and hence, multi-objective solutions are more widely used. One of the recently well-accepted computational algorithms is Multi-objective Particle Swarm Optimization (MOPSO). This is an easily implemented and high time performance nature-inspired approach; however, the best solutions are not found for archiving, solution updating, and fast convergence problems faced in certain cases. This study investigates the previously proposed solutions for creating diversity in using MOPSO and proposes using random immigrants approach. Application of the proposed solution is tested in four different sets using Generational Distance, Spacing, Error Ratio, and Run Time performance measures. The achieved results are statistically tested against mutation-based diversity for all four performance metrics. Advantages of this new approach will support the metaheuristic researchers.

Keywords Metaheuristics · Multi-objective optimization · Particle swarm optimization · Random immigrants

Introduction

Nature-inspired optimization methods have been used effectively to solve a wide variety of complex problems that consist of both single and multiple objective search domains. Among these methods, swarm intelligence is a promising research area. Introduced to solve single objective problems, Particle Swarm Optimization (PSO) [15] has attracted many researchers in metaheuristic optimization area, and started to gain prominence at solving multiple objective problems not more than 5 years after its introduction (see [27] for the first attempt on multi-objective optimization). This is because of the relative simplicity and the success as a single-objective optimizer, as well as high speed of convergence [4,22]. Furthermore, due to its population based nature, it enables to obtain a set of trade-off solutions in a single run, unlike the traditional techniques which employ a series of separate runs [36].

✉ Ali Nadi Ünal
anunal@hho.edu.tr

Gülgün Kayakutlu
kayakutlu@itu.edu.tr

1 Hezarfen Aeronautics and Space Technologies Institute, National Defense University, Istanbul, Turkey

2 Energy Institute, Istanbul Technical University, Istanbul, Turkey

However, there still exist three main issues to be considered in Multi-objective Particle Swarm Optimization (MOPSO): (1) archive maintenance, (2) process to update global best and individual best, and (3) solutions for local optima and premature convergence problems [11,13].

Maintaining an external archive, which is used to keep a historical record of non-dominated solutions in accordance with a quality measure, serves the main purpose of multi-objective optimization. Computational cost and memory size considerations cause keeping the size of external archive fixed seems more efficient [13,29]. While maintaining the external archive, to obtain a fairly distributed set of non-dominated solutions, employing a density measure in objective space is a straightforward approach. Strategies such as crowding distance [34,37], adaptive grid [5], clustering [33], maximin fitness [21], parallel cell coordinate system [13], and hypersurface contribution [35] can be used for maintaining the archive.

Regarding the update issue, the movement of a particle in MOPSO is affected by personal and global best selection (i.e., the selection of leaders). The selection of leaders is a crucial issue [41], and this selection directly affects the convergence and diversity attitudes, and effectiveness of the algorithm [48]. In other words, the balance between exploitation and exploration capabilities depends on the leader selection process. The trade-off between exploration and exploitation is critical to the performance of an evolutionary algorithm [31,47].

Although the fast convergence is an advantage for PSO or MOPSO, it becomes a drawback, if it is not controlled effectively. Fast convergence (premature convergence), especially for the earlier stages of the run, may cause particles to be "trapped in a similar local topology" [9]. As a consequence, it may not be possible to achieve a precise approximation to the true Pareto front [26]. Perturbation operator (mutation, disturbance) is a common practice to compensate for premature convergence, and maintaining diversity of the swarm along the optimization process [9,13].

This paper contributes to the literature by proposing the use of "*random immigrants*" approach, an effective method to promote diversity for MOPSO. Random immigrant approach has been developed to address maintaining diversity for genetic algorithms and proved to be beneficial [25,43]. It is based on a simple philosophy of replacing the worst or randomly selected particles from the swarm with randomly created particles. To the best of authors' knowledge, ours is the first study to use random immigrants approach for MOPSO.

Literature review

This section aims to give basic concepts and definitions on multi-objective optimization, and a short survey related to the fast convergence problem mentioned above.

Multi-objective optimization

Multiple objective optimization problems deal with at least two objective functions to be optimized. These objective functions are non-commensurable and competing. It means that they may be represented in different units, and they may have same level of importance comparatively. Assuming all the objective functions to be minimized, a multi-objective optimization problem can be defined as in [36]:

$$\text{Minimize } f(\mathbf{x}) := [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \quad (1)$$

Subject to

$$g_i(\mathbf{x}) \le 0 \quad i = 1, 2, \ldots, m, \quad (2)$$

$$h_j(\mathbf{x}) = 0 \quad j = 1, 2, \ldots, p, \quad (3)$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$ represents decision variables vector, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, 2, \ldots, k$ gives objective functions, and $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, p$ represents inequality and equality constraints, respectively. The desired solution is in the form of "trade-off" solutions between objective functions [34]. In other words, an improvement in one of the objective functions causes worsening for at least one of the others.

The most common two approaches to a multi-objective optimization problem are: (1) transforming the problem into a single objective one; (2) obtaining a set of trade-off solutions (preserving the problem as is) [17]. For the first case, simple additive weighting can be used, or all but one objective function can be moved to the constraint set. However, with the weighting method, not all Pareto-optimal solutions can be found for the problems that have non-convex objective or search spaces [3]. Additionally, many different weights may result the same single solution [3], and it can be very difficult to precisely and accurately select the weights [17]. When moving objective functions to the constraint set, it may be difficult to set right-hand side values for objective functions as constraints.

The main goal of a multi-objective optimization algorithm is to identify solutions in Pareto-optimal set [17,19]. Yet, all the elements of a Pareto optimal set may not be desirable or achievable [6,36], and the Pareto-optimal set can be infinite, while we have some space and time limitations. Therefore, it is desirable to obtain a set of solutions that represents the Pareto-optimal set as well as possible [17].

Real-world multi-objective optimization problems may be too complex to be solved by exact methods, such as linear programming and gradient search [50]. Population-based metaheuristic algorithms are accepted as effective computational solvers for multi-objective optimization problems. Due to their search capabilities through large spaces using populations, they are able to get some Pareto-optimal solutions in a single optimization run. Additionally, they are not effected by the shape of the Pareto-optimal front. Of these algorithms, MOPSO is a competitive one. Interested reader should refer [36] for a detailed explanation of a general MOPSO algorithm. Our approach omits the mutation phase of general MOPSO algorithm and adds random immigrants step after updating the leaders in an external archive.

Related work

Fieldsend and Singh [10] introduced a multi-objective algorithm based on the PSO and demonstrated the inclusion of stochastic turbulence variable. Using this new variable in MOPSO, they showed significant performance increases.

Mostaghim and Teich [28] proposed a method, called Sigma method for selecting the best local guide for each particle. They added a turbulence factor to the updated position of each particle in the swarm.

Xiao-hua et al. [42] proposed a modified PSO named Intelligent Particle Swarm Optimization (IPSO). They used a "clonal selection operator" to accelerate the approximation to optimum. One of the elements in this operator is called "clonal mutation", which helps to produce a solution set around Pareto optimal solutions.

Sierra and Coello [37] proposed a multi-objective particle swarm optimizer, which is based on Pareto dominance and uses of a crowding factor. They used uniform and non-uniform mutation schemes. In uniform mutation, the variability range allowed for each decision variable is kept constant over iterations, whilst in non-uniform mutation, this variability decreases over time.

Raquel and Naval [34] proposed an MOPSO algorithm which is called MOPSO-CD. They used crowding distance mechanism and mutation operator to maintain diversity. They performed mutation on the entire swarm initially and then rapidly decreased its coverage over time.

Peng and Zhang [32] proposed a decomposition based MOPSO. They applied the polynomia mutation on positions after they are calculated.

Izui et al. [14] proposed a multi-objective optimization method for structural problems based on MOPSO. They applied a mutation operator. In this operator, the probability of mutation decreases as the number of iterations increases, while mutation rate is fixed.

Agrawal et al. [1] proposed an interactive particle swarm metaheuristic for multi-objective optimization. They employed a mutation operator which is defined "self-adaptive mutation". This operator has some variation in probability according to the number of particles in the repository.

Padhye et al. [30] reviewed some proposals for guide selection in MOPSO and compared them with each other in terms of convergence, diversity, and computational times. They made a proposal named "velocity trigger" as a substitute for turbulence operator coupled with a boundary handling method. They reported that the new proposals were found to be effective for higher objective and higher parameter space problems.

Yen and Leong [44] proposed an MOPSO algorithm with dynamic multiple swarms. In the swarm growing strategy, they use uniform mutation operator with the mutation rate equal to one/number of dimensions in decision space.

Al Moubayed et al. [2] proposed an MOPSO algorithm that employs decomposition. Instead of mutation, they used an information exchange method that helps avoiding local optima without a need for applying any genetic operator.

Yen and Leong [45] proposed a constraint MOPSO which adopts a multi-objective constraint handling technique. They applied uniform and Gaussian operators. Uniform mutation encourage exploration and Gaussian mutation promotes exploitation. The frequency of applying the mutation operators depends on the feasibility ratio of the particles' personal best.

Daneshyari and Yen [7] proposed a cultural MOPSO which adapts the parameters of the MOPSO using the knowledge stored in various parts of the belief space. They applied a time-decaying mutation operator. The number of particles that undergo mutation, the range of mutation for each mutated particle, and the dimensions selected for mutation are regulated accordingly.

Mahmoodabadi et al. [23] modified the MOPSO in two stages. The first stage involves combining PSO with convergence and divergence operators. The second stage involves new leader selection method and adaptive elimination method which aims to limit the number of non-dominated solutions in the archive. They used the divergence operator as a simple controlled mutation.

Hu and Yen [12] proposed a method for density estimation for selecting leaders and maintaining the archive in MOPSO. They used "Parallel Cell Distance" between a solution and all other solutions in an archive after the archive is mapped from Cartesian Coordinate System into Parallel Cell Coordinate System. To perturb an article, they used Gaussian Mutation.

Leung et al. [20] presented a new algorithm that extends PSO to deal with multi-objective problems. Their first contribution is that the square root distance computation among particles for local best selection, and second is the procedure to update the archive members. They used mutation operator to enhance the exploratory ability of the algorithm.

Fan, Chang, and Chuang [9] proposed a multi-objective particle swarm optimizer which is constructed based on the concept of Pareto dominance taking both the diversified search and empirical movement strategies into account. They used polynomial mutation to maintaining the diversity of the particles along the optimization process.

Hu and Yen [13] proposed an integrated and adaptive MOPSO based on Parallel Cell Coordinate System (pccsAMOPSO). Their proposal includes a leader group, self-adaptive parameters, and perturbing operator for balancing convergence and diversity. They employed an elitism learning strategy with a Gaussian mutation as the perturbation operator.

Zhu et al. [49] presented a novel archive-guided MOPSO algorithm (AgMOPSO) where the leaders for velocity update are selected from an external archive. They also used an immune-based evolutionary strategy to evolve the external archive. They stated that this kind of updating scheme was verified to promote the convergence speed and keep the diversity.

Han et al. [11] proposed a variant of MOPSO, named Adaptive Gradient Multi-objective Particle Swarm Optimization (AGMOPSO). They used self-adaptive flight parameters mechanism to balance the convergence and diversity. They claimed that the proposed algorithm can find better spread solutions and has faster convergence to the true Pareto-optimal front.

Xiang et al. [41] proposed a many objective PSO (MaPSO). They suggested a new leader selection strategy. They kept multiple historical solutions from which the leader is selected for each particle. They also use linearly decreased parameter which promotes convergence initially and diversity later. It was shown that their proposed MaPSO is highly competitive or significantly superior to other algorithms. In another study on many objective PSO, Luo et al. [22] proposed an algorithm called IDMOPSO. They used a selection strategy for personal best to enhance the capability of local exploration. They also developed a multi-global best selection mechanism to balance convergence and diversity.

Pan et al. [31] proposed a diversity enhanced multi-objective particle swarm optimization called DEMPSO. In that study, analysis of particles' velocities is developed to assist variable clustering and elite selection. A diversity enhancing process based on the velocity analysis is carried out during the particles' evolution.

Current study intents to make a slight extension to [5]. Coello et al. [5] presented an approach in which Pareto-dominance is incorporated into standard PSO to handle problems with several objective functions. In that study, the movement of a particle is based on its own previous movements (personal best) and the movements of particles in a repository (i.e., leader is selected from an external archive of non-dominated solutions), as well. If the current position of a particle is better than the previous movements, current position is located as personal best. On the other hand, leader is selected randomly from the repository with respect to locations of non-dominated solutions from a hypercube. A more crowded hypercube has less chance to be selected. External archive and the positions of the solutions included in this archive are updated regularly at each iteration. A dynamic mutation operator is also employed in that study. Both the numbers of the particles which are subject to be applied mutation operator and the positions of a particle to be mutated decrease through iterations.

In the current study, we applied the same procedures like [5] except the mutation operator. Instead of mutation operator, random immigrants method is used for diversity preservation.

Random immigrants method

In the field of evolutionary algorithms, random immigrants method is known as an effective tool for diversity, especially for dynamic optimization [38,43]. This method is based on replacing some of the individuals (particles) with new ones. New individuals may be included in a complete random manner, or inclusion may be based on a memory scheme. The predecessors, or omitted individuals, can be selected randomly, or they can be determined with respect to a quality measure (e.g., fitness value or dominance relation).

Coello et al. [5] used mutation operator, such that the number of mutant individuals and the range of effect on decision variables decrease in a nonlinear fashion. In other words, all the particles are affected in the beginning with a high range. As the iteration number gets closer to the end, the number of mutants decreases to almost zero. Doing so, the algorithm gains a highly explorative and leveled (between exploration and exploitation) search capacity.

Algorithm 1 Random immigrants procedure

```
currentGen ← Current iteration number
totGen ← The number of iterations
decRate ← Decay rate
nPop ← The number of population
imigProb ← (1 − (currentGen − 1)/(totGen − 1))^(1/decRate) Immigration probability
if rand < imigProb then
    numImig ← (nPop ∗ (1 − (currentGen − 1)/(totGen − 1))^(1/decRate))/1.618) The number of random immigrants for current iteration
    Determine numImig particles randomly
    Create numImig random immigrants
    Omit determined particles
    Include random immigrants instead of omitted particles
end if
```

In a previous study [46], Martinez and Coello employed a re-initialization procedure to increase the diversity. In that study, a particle increases its age when it does not improve its personal position, and after exceeding a pre-defined age threshold, the particle is re-initialized. Our approach is totally different by means of re-initialization scheme.

Experimentation

Test problems

The first test problem was used by [8], and it is given in Eq. (4):

$$\begin{aligned} &\text{Minimize } f_1(x_1, x_2) = x_1 \\ &\text{Minimize } f_2(x_1, x_2) = \frac{g(x_2)}{x_1} \\ &\qquad \text{Subject to} \\ &\qquad g(x_2) = 2.0 - e^{-\left(\frac{x_2-0.2}{0.004}\right)^2} - 0.8e^{-\left(\frac{x_2-0.6}{0.4}\right)^2} \\ &\qquad 0.1 \le x_1, x_2 \le 1. \end{aligned} \tag{4}$$

The second test problem was used by [16], and given in Eq. (5):

$$\text{Maximize } f_1(x_1, x_2) = -x_1^2 + x_2$$
$$\text{Maximize } f_2(x_1, x_2) = \frac{1}{2}x_1 + x_2 + 1$$

Subject to

$$\frac{1}{6}x_1 + x_2 - \frac{13}{2} \leq 0$$
$$\frac{1}{2}x_1 + x_2 - \frac{15}{2} \leq 0$$
$$\frac{5}{x_1} + x_2 - 30 \leq 0$$
$$0 \leq x_1, x_2 \leq 7 \quad (5)$$

The third test problem was used by [18], and it is given in Eq. (6):

$$\text{Minimize } f_1(\mathbf{x}) = \sum_{i=1}^{2} -10e^{-0.2\sqrt{x_i^2+x_{i+1}^2}}$$
$$\text{Minimize } f_2(\mathbf{x}) = \sum_{i=1}^{3} \left(|x_i|^{0.8} + 5\sin(x_i^3)\right)$$
$$-5 \leq x_i \leq 5$$
$$i = 1, 2, 3 \quad (6)$$

Similar motivation is employed in the current study for random immigrants method. Immigration probability and the number of immigrants decrease with a nonlinear fashion. Random immigrants procedure is shown in Algorithm 1.

The fourth test problem is called “portfolio optimization problem” [24], and it is given in Eq. (7):

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{ij}$$
$$\text{Maximize} \sum_{i=1}^{N} x_i \mu_i$$

Subject to

$$\sum_{i=1}^{N} x_i = 1,$$
$$0 \leq x_i \leq 1,\ i = 1, \ldots, N. \quad (7)$$

Table 1 Values of parameters

Parameter	Value
Number of iterations	100
Swarm size	100
Repository size	100
W Inertia weight	0.5
C_1 Cognitive coefficient	1
C_2 Social coefficient	2
Number of grids	30
Mutation rate/decay rate	0.5
Number of runs	30

The data set for the first three test problems can be obtained from [39], while for the fourth one, it is available in [40].

Performance metrics

In this study, inspired by the analysis described in [5], three performance metrics are used for comparisons. These metrics are *generational distance* , *spacing*, and *error ratio*, and they are given in Eqs. (8)–(10).

$$D = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \quad (8)$$

where n is the number of vectors in the set of nondominated solutions found so far, and d_i is the Euclidean distance between each of these and the nearest member of the Pareto optimal set:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}, \quad (9)$$

where $d_i = \min_j(|f_1^i(\mathbf{x}) - f_1^j(\mathbf{x})| + |f_2^i(\mathbf{x}) - f_2^j(\mathbf{x})|)$, $i, j = 1, \ldots, n$, $i \neq j$, $\bar{d}$ is the mean of all d_i, and n is the number of nondominated vectors so far:

$$E = \frac{\sum_{i=1}^{n} e_i}{n}, \quad (10)$$

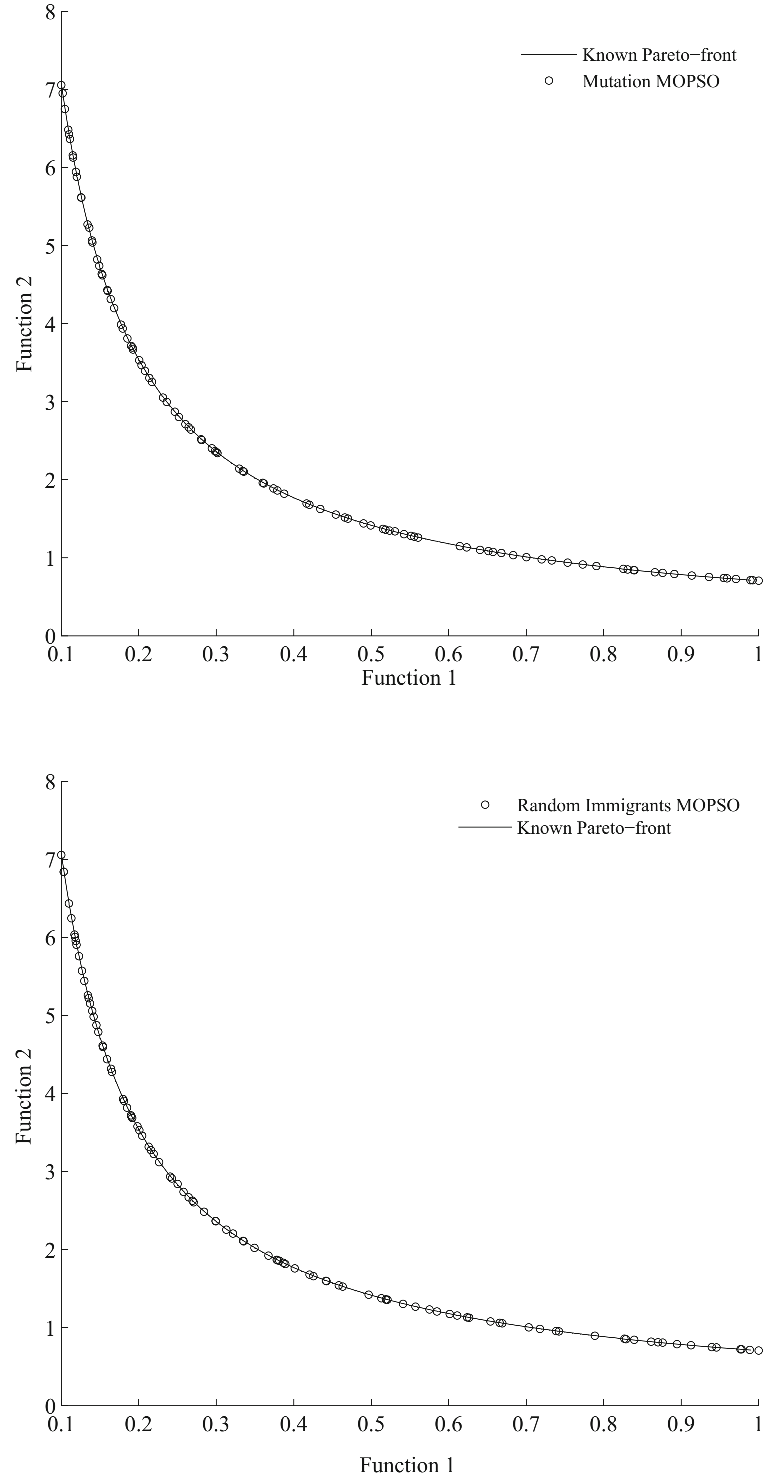
where n is the number of vectors in the current set of nondominated vectors available, $e_i = 0$ if the vector i is a member of the Pareto optimal set, and $e_i = 1$, otherwise. In addition to the above-mentioned metrics, run times are also evaluated to compare the speed of the two competing algorithms.

Hypothesis testing and parameters

To test how effective the proposed approach is, we compared two algorithms, namely MOPSO with mutation and MOPSO with random immigrants. We run both algorithms 30 times for each of the four problems. We compared both algorithms with respect to four performance metrics, both statistically and graphically. Statistical hypothesis testing (z test) is performed for comparison. The null hypothesis and the alternative hypothesis are stated as:

H_0: $\mu_1 = \mu_2$ *There is no significant difference between two approaches with respect to the related performance metric.*

Fig. 1 Pareto-fronts for the first test problem



H_1: $\mu_1 \neq \mu_2$ *There is a significant difference between two approaches with respect to the related performance metric.*

Here, μ_1 represents a mean value of 30 runs for respective performance metric for mutation MOPSO, and μ_2 represents a mean value of 30 runs for respective performance metric for random immigrants MOPSO. The parameters for both approach are given in Table 1.

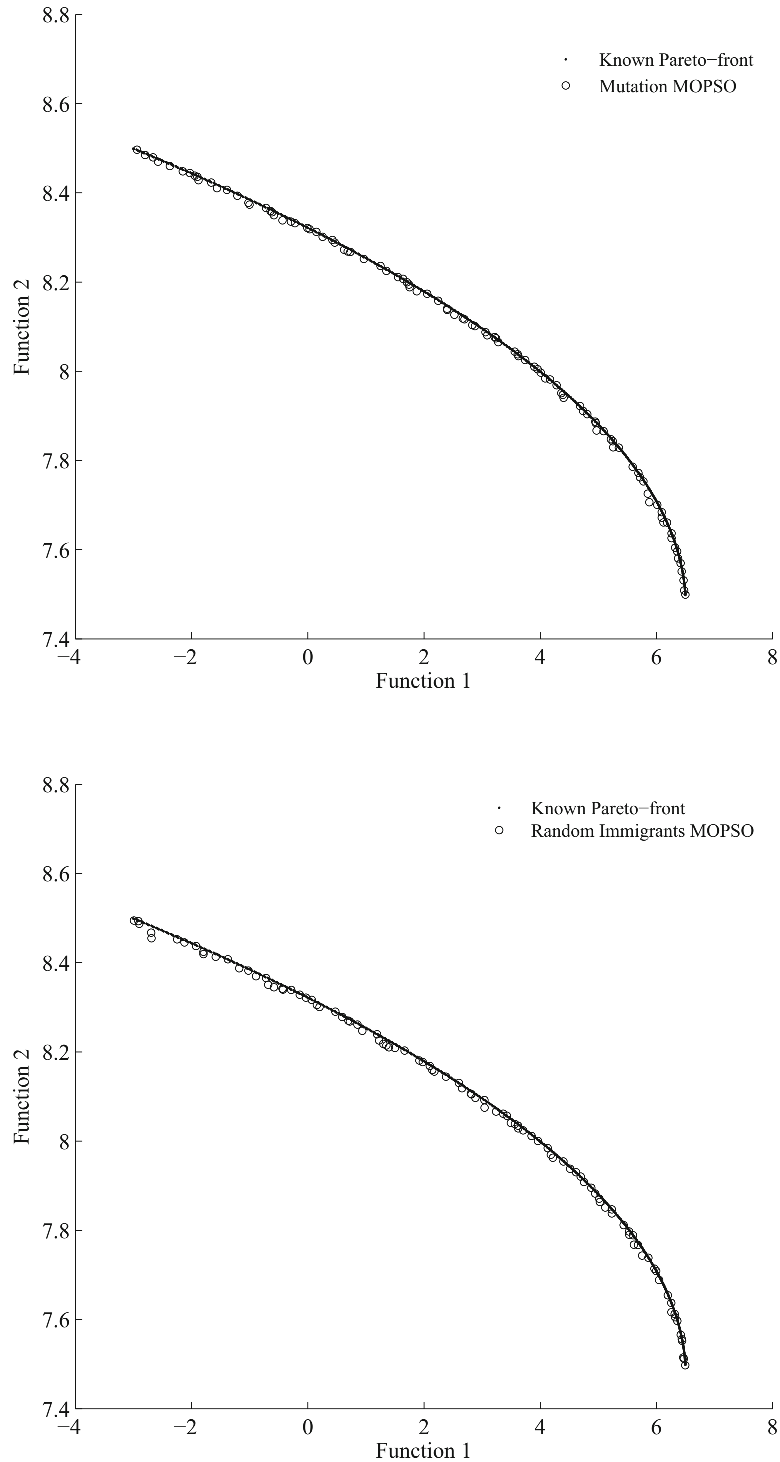


Fig. 2 Pareto-fronts for the second test problem

Results

Graphical comparison for all four test problems is given in Figs. 1, 2, 3 and 4. The results shown in these figures represent the best solutions with respect to the *D* performance metric. It can be seen that both mutation and random immigrants MOPSO produce a good approximation for the real Pareto-front for given test problems.

For each test problem, results for each run with respect to each performance metric are given in Tables 2, 4, 6, and 8, respectively. Based on these results, *z-test* is performed, and *z* values are given in Tables 3, 5, 7, and 9.

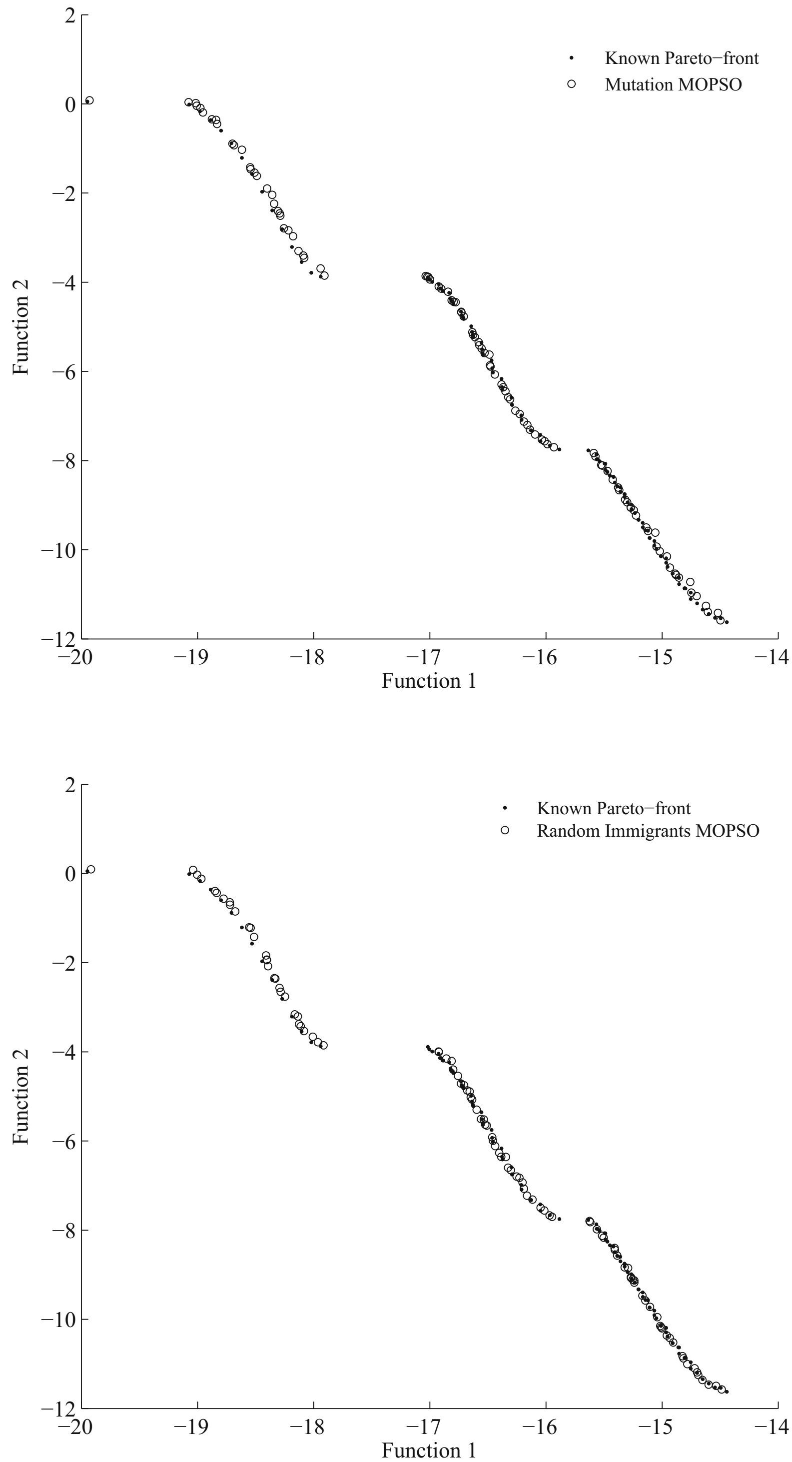


Fig. 3 Pareto-fronts for the third test problem

Results for the first test problem

According to the results given in Tables 2 and 3 for S performance metric null hypothesis can be rejected. It means that there is a significant difference between two approaches, with respect to the mean values of S. It seems that the mutation MOPSO outperforms random immigrants MOPSO for this performance metric.

Regarding the D and E performance metrics, there is not enough evidence to reject null hypothesis. It means that there is no significant difference between two approaches, with respect to the mean values of D and E. For the *time* performance metric, the null hypothesis can be rejected. It means

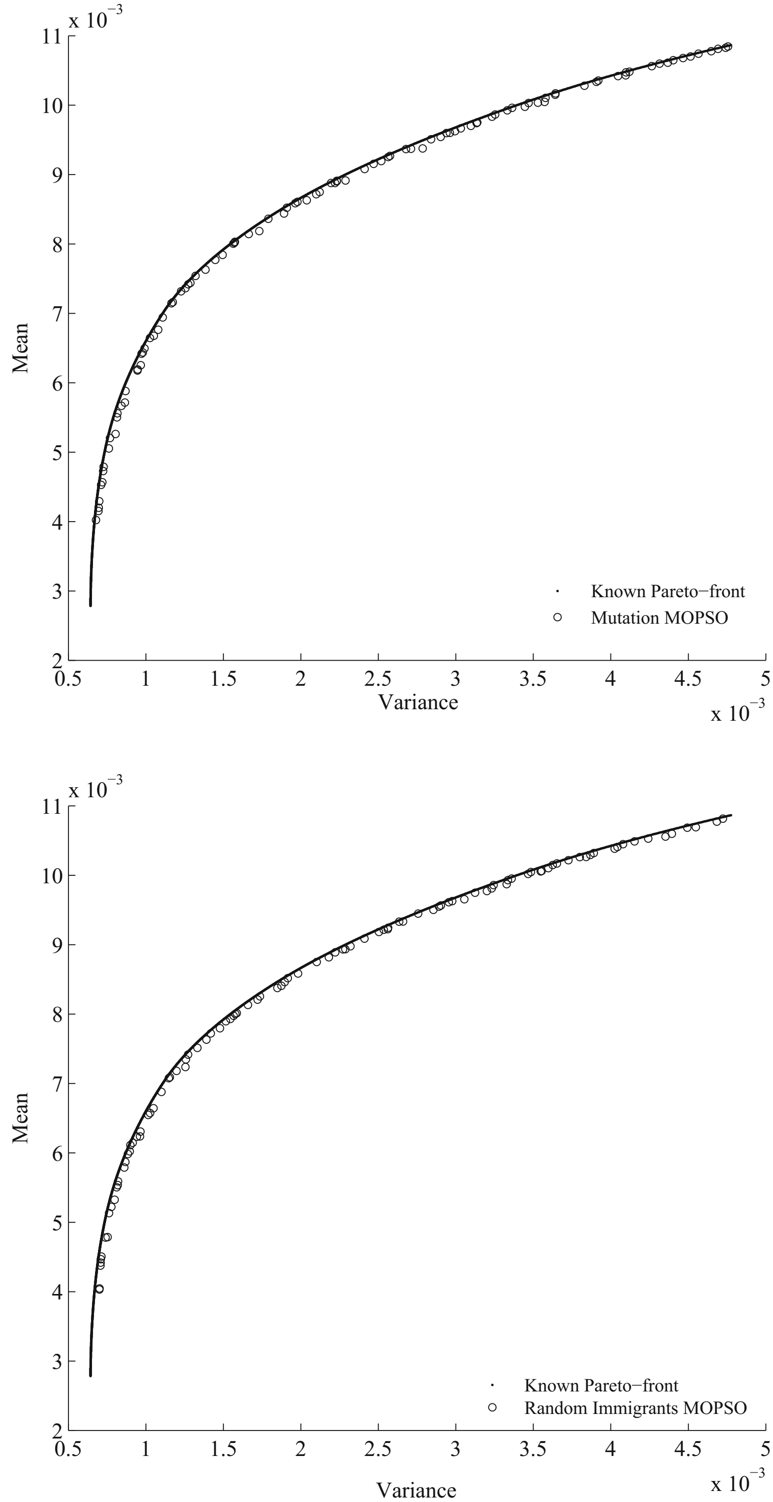


Fig. 4 Pareto-fronts for the fourth test problem

that there is a significant difference between the approaches, with respect to the run times of the algorithms. Random immigrants MOPSO seems to outperform mutation MOPSO method.

Results for the second test problem

According to the results given in Tables 4 and 5, for S, D, and E performance metrics, null hypothesis cannot be rejected. In other words, there is no significant difference between two approaches. Regarding the *time* performance metric, null

Table 2 Results for the first test problem

Run/statistic	Performance metrics							
	S		*D*		*E*		*T*	
	RI	Mut.	RI	Mut.	RI	Mut.	RI	Mut.
1	0.0426	0.0435	0.0038	0.0039	0.0200	0.0300	42.7821	46.3837
2	0.0429	0.0365	0.0043	0.0040	0.0100	0.0400	43.7493	48.4386
3	0.0401	0.0423	0.0044	0.0048	0.0100	0.0400	44.2675	51.7774
4	0.0443	0.0326	0.0038	0.0043	0.0200	0.0300	43.7091	48.6544
5	0.0455	0.0358	0.0046	0.0040	0.0300	0.0300	43.2256	46.3498
6	0.0374	0.0366	0.0044	0.0048	0.0300	0.0300	43.3359	48.6472
7	0.0431	0.0384	0.0042	0.0051	0.0100	0.0200	43.6625	48.7798
8	0.0444	0.0376	0.0044	0.0043	0.0300	0.0100	44.6642	47.8639
9	0.0436	0.0423	0.0043	0.0040	0.0200	0.0100	43.0536	46.9575
10	0.0374	0.0404	0.0047	0.0043	0.0400	0.0200	45.1330	48.7515
11	0.0331	0.0345	0.0048	0.0051	0.0100	0.0300	44.7500	48.9877
12	0.0346	0.0349	0.0043	0.0040	0.0300	0.0200	44.1840	49.5633
13	0.0327	0.0391	0.0043	0.0045	0.0300	0.0400	42.6265	47.4781
14	0.0372	0.0315	0.0040	0.0038	0.0200	0.0200	45.2132	49.0925
15	0.0432	0.0389	0.0044	0.0042	0.0200	0.0200	44.3408	47.6364
16	0.0416	0.0363	0.0044	0.0041	0.0200	0.0100	45.2360	50.4313
17	0.0385	0.0362	0.0044	0.0045	0.0300	0.0400	42.2915	47.4614
18	0.0357	0.0352	0.0042	0.0044	0.0300	0.0300	44.3946	48.3598
19	0.0453	0.0431	0.0043	0.0041	0.0300	0.0100	45.7289	46.5018
20	0.0419	0.0386	0.0046	0.0043	0.0300	0.0200	44.4128	47.3061
21	0.0433	0.0366	0.0042	0.0046	0.0100	0.0400	42.8689	46.8911
22	0.0481	0.0331	0.0043	0.0046	0.0300	0.0200	43.5025	47.4834
23	0.0422	0.0435	0.0045	0.0045	0.0300	0.0000	43.0032	47.7383
24	0.0385	0.0313	0.0042	0.0041	0.0100	0.0200	46.0204	47.9371
25	0.0459	0.0396	0.0041	0.0049	0.0500	0.0100	43.5170	48.5421
26	0.0339	0.0290	0.0047	0.0044	0.0400	0.0300	44.2943	47.6873
27	0.0416	0.0323	0.0048	0.0043	0.0300	0.0300	44.0153	47.5622
28	0.0388	0.0409	0.0044	0.0040	0.0400	0.0100	42.5261	47.1278
29	0.0355	0.0283	0.0044	0.0049	0.0300	0.0100	45.7032	47.0328
30	0.0390	0.0425	0.0048	0.0052	0.0300	0.0500	44.3875	48.6581
Min.	0.0327	0.0283	0.0038	0.0038	0.0100	0.0000	***42.2915***	46.3498
Max.	0.0481	0.0435	***0.0048***	0.0052	***0.0500***	0.0500	***46.0204***	51.7774
Mean	0.0404	0.0370	***0.0044***	0.0044	0.0257	0.0240	***44.0200***	48.0694
Std. dev.	0.0041	0.0043	0.0003	0.0004	0.0104	0.0122	1.0076	1.1910

Table 3 Statistical hypothesis testing for the first test problem

Performance metric	*z* value	H_0
S	3.08	Reject
D	− 0.4496	Not enough evidence to reject
E	0.5694	Not enough evidence to reject
T	− 14.2174	Reject

$\alpha < 0.05$

Table 4 Results for the second test problem

Run/statistic	Performance metrics							
	S		*D*		*E*		*T*	
	RI	Mut.	RI	Mut.	RI	Mut.	RI	Mut.
1	0.0491	0.0374	0.0113	0.0139	0.1100	0.1200	38.6895	38.1807
2	0.0484	0.0515	0.0119	0.0344	0.1800	0.1500	39.3091	37.9143
3	0.0648	0.0480	0.0176	0.0267	0.1700	0.0900	36.5373	40.3929
4	0.0539	0.0771	0.0233	0.0142	0.1000	0.1300	37.6547	38.1014
5	0.1574	0.1580	0.0503	0.1526	0.0900	0.1200	38.1928	38.2802
6	1.5723	0.1409	0.1738	0.0905	0.1100	0.1100	36.6740	38.3075
7	0.0515	0.9726	0.0408	0.1995	0.1700	0.0600	37.7982	36.8344
8	0.2944	0.0542	0.0634	0.0246	0.1300	0.0700	38.4847	54.4839
9	0.0504	0.0576	0.0084	0.0159	0.1400	0.1900	39.9653	54.3228
10	0.1311	0.0652	0.1657	0.0404	0.1400	0.1700	34.6981	38.9263
11	0.0411	0.0811	0.0124	0.0227	0.1200	0.0900	37.0576	36.7245
12	0.1304	0.1706	0.0361	0.0255	0.1900	0.1500	36.5183	38.0513
13	0.1097	0.3508	0.0224	0.0768	0.2000	0.0900	37.5736	39.4193
14	0.9711	0.0460	0.1132	0.0679	0.1500	0.1100	37.4087	39.1830
15	0.0516	0.7488	0.0218	0.0924	0.1000	0.1300	39.1586	38.4872
16	0.4443	0.1697	0.1241	0.0271	0.1700	0.1600	37.9423	38.1918
17	0.0855	0.1534	0.0408	0.0239	0.1200	0.1500	38.7080	39.3768
18	0.0466	0.0505	0.0231	0.0074	0.1100	0.1100	38.2719	37.9889
19	0.5780	0.7758	0.0693	0.0940	0.0900	0.1100	37.0737	37.6973
20	0.7394	0.0577	0.2124	0.0581	0.1500	0.1400	36.2117	38.6942
21	0.0447	0.7446	0.0094	0.1712	0.1800	0.1800	37.2069	37.6676
22	0.0603	0.1034	0.0120	0.0245	0.1100	0.1300	37.2697	38.9032
23	0.0462	0.1047	0.0122	0.0794	0.1900	0.1300	37.3081	39.4277
24	0.2703	0.0457	0.0463	0.0145	0.1000	0.0900	38.5350	39.0234
25	0.0545	0.0439	0.0301	0.0166	0.1500	0.1200	37.1355	39.7306
26	0.0429	0.1315	0.0097	0.1110	0.1000	0.1600	39.3758	36.6069
27	0.0489	0.0920	0.0109	0.0276	0.1300	0.1700	38.8031	38.8613
28	0.0829	0.0541	0.0509	0.0087	0.1100	0.1800	38.0753	38.8249
29	0.2253	0.4364	0.0429	0.1417	0.1400	0.0900	38.9690	37.4339
30	0.0833	0.0465	0.0232	0.0153	0.1300	0.1200	38.9606	39.9278
Min.	0.0411	0.0374	0.0084	0.0074	0.0900	0.0600	***34.6981***	36.6069
Max.	1.5723	0.9726	0.2124	0.1995	0.2000	0.1900	***39.9653***	54.4839
Mean	0.2210	0.2023	***0.0497***	0.0573	0.1360	0.1273	***37.8522***	39.5322
Std. dev.	0.3408	0.2605	0.0540	0.0530	0.0329	0.0338	1.1341	4.1443

Table 5 Statistical hypothesis testing for the second test problem

Performance metric	*z* value	H_0
S	0.2387	Not enough evidence to reject
D	− 0.5539	Not enough evidence to reject
E	1.007	Not enough evidence to reject
T	− 2.1514	Reject

$\alpha < 0.05$

Table 6 Results for the third test problem

Run/statistic	Performance metrics							
	S		*D*		*E*		*T*	
	RI	Mut.	RI	Mut.	RI	Mut.	RI	Mut.
1	0.1086	0.1080	0.0589	0.0558	0.0000	0.0000	30.7471	31.9891
2	0.0999	0.1021	0.0595	0.0579	0.0000	0.0000	32.7430	31.2913
3	0.0694	0.1115	0.0590	0.0579	0.0400	0.0000	31.3615	30.7032
4	0.0887	0.1043	0.0523	0.0549	0.0300	0.0000	31.4082	31.5288
5	0.1169	0.1165	0.0512	0.0591	0.0000	0.0000	31.2170	29.6907
6	0.0595	0.0651	0.0561	0.0602	0.0000	0.0000	31.2949	31.0227
7	0.0735	0.0953	0.0619	0.0519	0.0200	0.0000	31.8885	30.3834
8	0.0955	0.0577	0.0578	0.0604	0.0000	0.0300	30.8581	31.2546
9	0.1176	0.1085	0.0579	0.0584	0.0000	0.0100	30.4882	30.3319
10	0.1047	0.0590	0.0532	0.0563	0.0000	0.0200	31.8226	31.3167
11	0.1056	0.1011	0.0531	0.0602	0.0000	0.0000	30.8987	30.2592
12	0.1006	0.1155	0.0554	0.0542	0.0000	0.0000	30.8969	30.8225
13	0.1017	0.0573	0.0597	0.0570	0.0000	0.0300	30.6360	30.0280
14	0.1069	0.1062	0.0587	0.0595	0.0000	0.0000	32.1974	30.5088
15	0.0622	0.1063	0.0568	0.0599	0.0000	0.0000	31.1471	30.7564
16	0.0952	0.1072	0.0563	0.0528	0.0100	0.0000	31.4530	31.2568
17	0.1009	0.0987	0.0544	0.0581	0.0000	0.0100	31.6583	31.3700
18	0.0535	0.1076	0.0569	0.0547	0.0000	0.0000	31.9432	31.1027
19	0.0668	0.0657	0.0546	0.0583	0.0000	0.0000	30.9934	31.1797
20	0.0972	0.1110	0.0503	0.0632	0.0000	0.0000	31.4585	30.0472
21	0.1124	0.0644	0.0601	0.0545	0.0000	0.0000	31.6670	29.9122
22	0.1111	0.0725	0.0582	0.0598	0.0100	0.0200	31.1748	30.2498
23	0.0592	0.1118	0.0594	0.0567	0.0200	0.0000	31.5038	31.4885
24	0.0688	0.0562	0.0568	0.0595	0.0000	0.0100	31.5334	31.7842
25	0.1021	0.1056	0.0535	0.0558	0.0000	0.0000	31.7340	31.4312
26	0.1066	0.1073	0.0544	0.0544	0.0000	0.0000	31.5341	30.5024
27	0.0703	0.1016	0.0542	0.0601	0.0000	0.0100	31.6510	31.0848
28	0.0917	0.0611	0.0657	0.0593	0.0100	0.0300	31.8327	30.6619
29	0.1258	0.0951	0.0512	0.0585	0.0000	0.0000	30.8778	31.0640
30	0.1013	0.0598	0.0557	0.0573	0.0000	0.0000	31.4873	30.5123
Min.	***0.0535***	0.0562	***0.0503***	0.0519	***0.0000***	0.0000	30.4882	29.6907
Max.	0.1258	0.1165	0.0657	0.0632	0.0400	0.0300	32.7430	31.9891
Mean	0.0925	0.0913	***0.0564***	0.0575	***0.0047***	0.0057	31.4036	30.8512
Std. dev.	0.0202	0.0219	0.0034	0.0026	0.0101	0.0101	0.4895	0.5828

Table 7 Statistical hypothesis testing for the third test problem

Performance metric	*z* value	H_0
S	0.2069	Not enough evidence to reject
D	1.4085	Not enough evidence to reject
E	− 0.3844	Not enough evidence to reject
T	3.9755	Reject

$\alpha < 0.05$

Table 8 Results for the fourth test problem

Run/statistic	Performance metrics							
	S		*D*		*E*		*T*	
	RI	Mut.	RI	Mut.	RI	Mut.	RI	Mut.
1	6.22E−05	4.72E−05	3.62E−05	3.77E−05	0.44	0.35	37.449	38.873
2	4.48E−05	4.23E−05	3.90E−05	3.7E−05	0.4	0.29	36.834	37.131
3	4.24E−05	4.48E−05	3.18E−05	3.13E−05	0.33	0.35	38.478	37.947
4	4.14E−05	4.37E−05	3.49E−05	3.63E−05	0.38	0.32	37.497	37.047
5	4.21E−05	4.41E−05	3.39E−05	3.84E−05	0.41	0.33	37.722	37.162
6	4.75E−05	4.51E−05	3.55E−05	4.13E−05	0.33	0.34	37.098	37.436
7	4.26E−05	4.06E−05	3.53E−05	2.69E−05	0.32	0.35	37.250	37.927
8	5.03E−05	5.47E−05	3.28E−05	4.5E−05	0.35	0.29	36.809	38.579
9	5.1E−05	4.34E−05	3.97E−05	4.3E−05	0.39	0.31	37.178	38.249
10	5.33E−05	4.21E−05	3.55E−05	3.37E−05	0.34	0.39	42.377	37.440
11	4.35E−05	4.96E−05	3.09E−05	2.84E−05	0.41	0.37	43.261	37.812
12	5.7E−05	4.71E−05	3.94E−05	3.31E−05	0.24	0.35	36.158	37.626
13	4.84E−05	3.51E−05	3.85E−05	2.79E−05	0.37	0.31	36.42133	37.189
14	4.53E−05	4.02E−05	3.48E−05	3.28E−05	0.31	0.34	37.196	37.956
15	5.41E−05	4.85E−05	3.36E−05	3.02E−05	0.36	0.36	36.787	37.476
16	4.18E−05	5.48E−05	3.61E−05	3.38E−05	0.41	0.34	37.164	37.412
17	4.86E−05	5.06E−05	3.26E−05	3.15E−05	0.35	0.32	37.164	36.968
18	4.14E−05	4.51E−05	3.94E−05	4.19E−05	0.35	0.29	37.159	37.190
19	4.38E−05	5.23E−05	3.41E−05	3.74E−05	0.36	0.38	36.597	37.207
20	4.6E−05	4.28E−05	4.30E−05	3.98E−05	0.37	0.35	37.350	36.977
21	4.22E−05	4.53E−05	3.87E−05	2.63E−05	0.37	0.31	37.392	38.036
22	4.21E−05	4.49E−05	3.07E−05	3.6E−05	0.41	0.38	39.429	37.074
23	4.6E−05	5.36E−05	3.75E−05	4.05E−05	0.34	0.31	37.151	37.769
24	4.46E−05	4.82E−05	3.89E−05	3.84E−05	0.32	0.28	36.943	37.436
25	4.11E−05	5.11E−05	3.80E−05	4.48E−05	0.37	0.34	37.255	38.352
26	4.69E−05	5.12E−05	3.36E−05	3.81E−05	0.33	0.42	39.896	37.806
27	4.93E−05	5.22E−05	3.63E−05	4.14E−05	0.37	0.32	38.376	37.567
28	5.93E−05	4.09E−05	3.53E−05	3.55E−05	0.33	0.27	37.505	37.400
29	3.91E−05	4.03E−05	3.29E−05	2.62E−05	0.32	0.36	37.812	38.072
30	7.45E−05	4.37E−05	3.66E−05	3.76E−05	0.31	0.3	36.186	37.293
Min.	3.91E−05	3.51E−05	3.08E−05	2.62E−05	***0.24***	0.27	***36.158***	36.968
Max.	7.45E−05	5.48E−05	***4.30E−05***	4.5E−05	0.44	0.42	43.261	38.874
Mean	4.77E−05	4.62E−05	3.59E−05	3.57E−05	0.35	0.33	37.730	37.614
Std. dev.	7.63E−06	4.9E−06	2.93E−06	5.45E−06	0.04	0.03	1.608	0.491

hypothesis can be rejected. There is a significant difference between two approaches, and random immigrants MOPSO outperforms mutation MOPSO algorithm.

Table 9 Statistical hypothesis testing for the fourth test problem

Performance metric	*z* value	H_0
S	0.9411	Not enough evidence to reject
D	0.1451	Not enough evidence to reject
E	2.2727	Reject
T	0.3786	Not enough evidence to reject

$\alpha < 0.05$

Results for the third test problem

According to the results given in Tables 6 and 7, null hypothesis cannot be rejected. It means that there is no significant difference between two approaches with respect to these performance metrics. Regarding the *time* performance metric, null hypothesis can be rejected, and the mutation MOPSO outperforms the random immigrants MOPSO.

Results for the fourth test problem

According to the results given in Tables 8 and 9, for S and D performance metrics, null hypothesis cannot be rejected. It means that there is no significant difference between two approaches with respect to these performance metrics. Nevertheless, for E performance metric, null hypothesis can be rejected. It means that there is a significant difference between two approaches with respect to E performance metric, and mutation MOPSO approach outperforms random immigrants MOPSO approach. Regarding the *time* performance metric, null hypothesis cannot be rejected. In other words, there is no significant difference between two approaches with respect to the running times.

Conclusion

Computational Intelligence is developed to create solutions for the real-life complex problems for which linear, nonlinear, or stochastic models could not propose a remedy. Nature-inspired metaheuristic algorithms take an improving importance in computational intelligence. The most important issue in metaheuristics is to balance the exploitation and the exploration depending on the problem studied. When the problem has a single objective, the hybridized algorithms provide solutions, but it is more difficult when a multi objective analysis is made.

This paper is original to develop MOPSO using Random Immigrants, which has enlarged the diversity of solutions remarkably. Application of the proposed algorithm on four well-accepted data sets has shown the benefits of the solution for four performance measures, process timing, Generational Distance, Spacing, and Error Ratio. Statistical experiments are constructed to compare the Mutation Approach (that gives better than other existing solutions) and the Random Immigrants approach for diversity. Results show that Random Immigrants approach is faster in providing the solution in most of the cases and as good as mutation approach in the others.

Implementing the proposed algorithm in a real-life case and to compare the results (for both real-life cases and test suits) with the state-of-the-art variants of the MOPSO will be the extension of this study. Archive maintenance and local optima problems faced in using MOPSO worth new studies, which will be further analysis in our team. Having observed the timing performances of MOPSO with Random Immigrants, we would also like to recommend the application of this method in Big Data handling.

This paper will open a new dimension for the MOPSO researchers and provide a new tool for computational intelligence application.

References

1. Agrawal S, Dashora Y, Tiwari MK, Son YJ (2008) Interactive particle swarm: a pareto-adaptive metaheuristic to multiobjective optimization. IEEE Trans Syst Man Cybern Part A Syst Hum 38(2):258–277
2. Al Moubayed N, Petrovski A, McCall J (2010) A novel smart multi-objective particle swarm optimisation using decomposition. Springer, Berlin Heidelberg, pp 1–10
3. Baltar AM, Fontane DG (2006) A generalized multiobjective particle swarm optimization solver for spreadsheet models: application to water quality. Hydrol Days 1–12
4. Coello CAC, Lechuga MS (2002) Mopso: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 congress on evolutionary computation, 2002. CEC '02, vol 2, pp 1051–1056
5. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279
6. Coello Coello CA, González Brambila S, Figueroa Gamboa J, Castillo Tapia MG, Hernández Gómez R (2019) Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. Complex Intell Syst. https://doi.org/10.1007/s40747-019-0113-4
7. Daneshyari M, Yen GG (2011) Cultural-based multiobjective particle swarm optimization. IEEE Trans Syst Man Cybern Part B (Cybernetics) 41(2):553–567
8. Deb K (1999) Multi-objective genetic algorithms:problem difficulties and construction of test problems. Evol Comput 7:205–230

9. Fan SKS, Chang JM, Chuang YC (2015) A new multi-objective particle swarm optimizer using empirical movement and diversified search strategies. Eng Optim 47(6):750–770
10. Fieldsend JE, Singh S (2002) A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In: 2002 UK workshop on computational intelligence, Birmingham, UK, 2–4 September 2002, pp. 37–44
11. Han H, Lu W, Zhang L, Qiao J (2018) Adaptive gradient multiobjective particle swarm optimization. IEEE Trans Cybern 48(11):3067–3079. https://doi.org/10.1109/TCYB.2017.2756874
12. Hu W, Yen GG (2013) Density estimation for selecting leaders and mantaining archive in mopso. In: 2013 IEEE congress on evolutionary computation, pp 181–188
13. Hu W, Yen GG (2015) Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system. IEEE Trans Evol Comput 19(1):1–18
14. Izui K, Nishiwaki S, Yoshimura M, Nakamura M, Renaud JE (2008) Enhanced multiobjective particle swarm optimization in combination with adaptive weighted gradient-based searching. Eng Optim 40(9):789–804
15. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE international conference on neural networks, 1995. Proceedings, vol 4, pp 1942–1948
16. Kita H, Yabumoto Y, Mori N, Nishikawa Y (1996) Multi-objective optimization by means of the thermodynamical genetic algorithm. In: Proceedings of the 4th international conference on parallel problem solving from nature, PPSN IV. Springer, London, pp 504–512
17. Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: a tutorial. Reliab Eng Syst Saf 91(9):992–1007. Special Issue—Genetic Algorithms and ReliabilitySpecial Issue—Genetic Algorithms and Reliability
18. Kursawe F (1991) A variant of evolution strategies for vector optimization. In: Proceedings of the 1st workshop on parallel problem solving from nature, PPSN I. Springer, London, pp 193–197
19. Lalwani S, Singhal S, Kumar R, Gupta N (2013) A comprehensive survey: applications of multi-objective particle swarm optimization (mopso) algorithm. Trans Combin 2(1):39–101
20. Leung MF, Ng SC, Cheung CC, Lui AK (2014) A new strategy for finding good local guides in mopso. In: 2014 IEEE congress on evolutionary computation (CEC), pp 1990–1997
21. Li X (2004) Better spread and convergence: particle swarm multiobjective optimization using the maximin fitness function. Springer, Berlin Heidelberg, pp 117–128
22. Luo J, Huang X, Li X, Gao K (2019) A novel particle swarm optimizer for many-objective optimization. In: 2019 IEEE congress on evolutionary computation (CEC), pp 958–965. https://doi.org/10.1109/CEC.2019.8790343
23. Mahmoodabadi MJ, Bagheri A, Nariman-zadeh N, Jamali A (2012) A new optimization algorithm based on a combination of particle swarm optimization, convergence and divergence operators for single-objective and multi-objective problems. Eng Optim 44(10):1167–1186
24. Markowitz H (1952) Portfolio selection. J Fin 7(1):77–91
25. Mavrovouniotis M, Yang S (2013) Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. Appl Soft Comput 13(10):4023–4037
26. Meza J, Espitia H, Montenegro C, Giménez E, González-Crespo R (2017) Movpso: Vortex multi-objective particle swarm optimization. Appl Soft Comput 52:1042–1057. https://doi.org/10.1016/j.asoc.2016.09.026
27. Moore J, Chapman R, Dozier G (2000) Multiobjective particle swarm optimization. In: Proceedings of the 38th annual on southeast regional conference, ACM-SE 38. ACM, New York, pp 56–57
28. Mostaghim S, Teich J (2003) Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In: Swarm intelligence symposium, 2003. SIS '03. Proceedings of the 2003 IEEE, pp 26–33
29. Padhye N (2009) Comparison of archiving methods in multiobjectiveparticle swarm optimization (mopso): Empirical study. In: Proceedings of the 11th annual conference on genetic and evolutionary computation, GECCO '09. ACM, New York, pp 1755–1756
30. Padhye N, Branke J, Mostaghim S (2009) Empirical comparison of mopso methods: guide selection and diversity preservation. In: 2009 IEEE congress on evolutionary computation, pp 2516–2523
31. Pan A, Wang L, Guo W, Wu Q (2018) A diversity enhanced multiobjective particle swarm optimization. Inf Sci 436–437:441–465. https://doi.org/10.1016/j.ins.2018.01.038
32. Peng W, Zhang Q (2008) A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. In: 2008 IEEE international conference on granular computing, pp 534–537
33. Pulido GT, Coello Coello CA (2004) Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. Springer, Berlin, pp 225–237
34. Raquel CR, Naval Jr PC (2005) An effective use of crowding distance in multiobjective particle swarm optimization. In: Proceedings of the 7th annual conference on genetic and evolutionary computation, GECCO '05. ACM, New York, pp 257–264
35. Scheepers C, Engelbrecht AP (2017) Vector evaluated particle swarm optimization: The archive's influence on performance. In: 2017 IEEE congress on evolutionary computation (CEC), pp 565–572. https://doi.org/10.1109/CEC.2017.7969361
36. Sierra MR, Coello CAC (2006) Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int J Comput Intell Res 2(3):287–308
37. Sierra MR, Coello Coello CA (2005) Improving pso-based multiobjective optimization using crowding, mutation and ϵ-dominance. In: Proceedings of the third international conference on evolutionary multi-criterion optimization, EMO'05. Springer, Berlin, Heidelberg, pp 505–519
38. Unal AN (2013) A Genetic Algorithm for the Multiple Knapsack Problem in Dynamic Environment. In: World congress on engineering and computer science, WCECS 2013, vol II, Lecture Notes in Engineering and Computer Science, pp 1162–1167
39. Url-1: http://delta.cs.cinvestav.mx/~ccoello/EMOO/testfuncs/ (2016). Accessed 26 July 2016
40. Url-2: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html (2016). Accessed 26 July 2016
41. Xiang Y, Zhou Y, Chen Z, Zhang J (2018) A many-objective particle swarm optimizer with leaders selected from historical solutions by using scalar projections. IEEE Trans Cybern 1–14. https://doi.org/10.1109/TCYB.2018.2884083
42. Xiao-hua Z, Hong-yun M, Li-cheng J (2005) Intelligent particle swarm optimization in multiobjective optimization. In: 2005 IEEE congress on evolutionary computation, vol 1, pp 714–719
43. Yang S (2005) Memory-based immigrants for genetic algorithms in dynamic environments. In: Proceedings of the 7th annual conference on genetic and evolutionary computation, GECCO '05, pp 1115–1122. ACM, New York, NY, USA
44. Yen GG, Leong WF (2009) Dynamic multiple swarms in multiobjective particle swarm optimization. IEEE Trans Syst Man Cybern Part A Syst Hum 39(4):890–911
45. Yen GG (2010) Leong, W.F.: Constraint handling procedure for multiobjective particle swarm optimization. In: IEEE congress on evolutionary computation, pp 1–8
46. Zapotecas Martínez S, Coello Coello CA (2011) A multi-objective particle swarm optimizer based on decomposition. In: Proceedings of the 13th annual conference on genetic and evolutionary computation, GECCO '11. ACM, New York, pp 69–76. DOI https://doi.org/10.1145/2001576.2001587

47. Zhang H, Sun J, Liu T, Zhang K, Zhang Q (2019) Balancing exploration and exploitation in multiobjective evolutionary optimization. Inf Sci 497:129–148. https://doi.org/10.1016/j.ins.2019.05.046
48. Zhao SZ, Suganthan PN (2011) Two-lbests based multi-objective particle swarm optimizer. Eng Optim 43(1):1–17
49. Zhu Q, Lin Q, Chen W, Wong K, Coello Coello CA, Li J, Chen J, Zhang J (2017) An external archive-guided multiobjective particle swarm optimization algorithm. IEEE Trans Cybern 47(9):2794–2808. https://doi.org/10.1109/TCYB.2017.2710133
50. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195