Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

Ions motion optimization algorithm for multiobjective optimization problems

Hitarth Buch[a,b*] **and Indrajit Trivedi**[c]

[a]*Research Scholar, Gujarat Technological University, Ahmedabad*
[b]*Department of Electrical Engineering, Government Engineering College, Rajkot, Gujarat, India*
[c]*Department of Power Electronics, Vishwakarma Government Engineering College, Chandkheda, Gujarat, India*

C H R O N I C L E

Article history:
Received October 28, 2020
Received in revised format: October 29, 2020
Accepted November 17 2020
Available online November 17, 2020

Keywords:
Multiobjective optimization
Ions Motion Algorithm
Archive

A B S T R A C T

This paper offers a novel multiobjective approach – Multiobjective Ions Motion Optimization (MOIMO) algorithm stimulated by the movements of ions in nature. The main inspiration behind this approach is the force of attraction and repulsion between anions and cations. A storage and leader selection strategy is combined with the single objective Ions Motion Optimization (IMO) approach to estimate the Pareto optimum front for multiobjective optimization. The proposed method is applied to 18 different benchmark test functions to confirm its efficiency in finding optimal solutions. The outcomes are compared with three novel and well-accepted techniques in the literature using five performance parameters quantitatively and obtained Pareto fronts qualitatively. The comparison proves that MOIMO can approximate Pareto optimal solutions with good convergence and coverage with minimum computational time.

1. Introduction

The optimization process looks for finding the minimum or maximum value for single or multiple objectives. Multiobjective optimization refers to optimizing numerous objectives which are often conflicting in nature. Such problems are found in engineering, mathematics, economics, agriculture, politics, information technology, etc. Also, sometimes, the truly optimum solution may not be available at all. In such cases, compromise and estimates are frequently required. Multiobjective optimization is much more complicated than single-objective optimization because of the existence of multiple optimum solutions. At large, all solutions are conflicting, and hence, a group of non-dominated solutions is required to be found out to approximate the true pareto front. Heuristic algorithms are derivative-free solution approaches as they do not use gradient descent to determine the global optimal. Metaheuristic approaches treat the problem as a black box for given inputs and outputs. Problem variables are inputs while objectives are outputs. Many competent metaheuristic approaches were proposed in the past to solve the multiobjective optimization problem. A heuristic approach starts problem optimization by creating an arbitrary group of initial solutions. Every candidate solution is evaluated, objective values are observed, and based on the outputs, the candidate solutions are modified/changed/combined/evolved. This process is continued until the end criteria are met.

There are various difficulties associated while solving the problem using heuristics. Even optimization problems have diverse characteristics. Some of the challenges are constraints, uncertainty, multiple and many objectives, dynamicity. Over a while, global optimum value changes in dynamic problems. Hence, the heuristic approach should be furnished with a suitable operator to keep track of such changes so that the global optimum is not lost. Heuristic approaches should also be fault-tolerant to deal with uncertainty effectively. Constraints divide the search space into viable and unviable solutions. The heuristic approach should be able to discard the unsustainable solution and ultimately discover the best optimum

* Corresponding author.
E-mail address: hitarth_buch_020@gtu.edu.in (H. Buch)

doi: 10.5267/j.dsl.2020.12.001

solution. Researchers have also proposed surrogate models to reduce computational efforts for computationally expensive functions. The idea of a Pareto dominance operator is introduced to compare more than one objective. The heuristic approach should be able to find all the best Pareto solutions. The proper mechanism should be incorporated with heuristic approaches to deal with multiobjective problems. Archive or storage is necessary to stock the non-dominated solutions found through optimization. Quantity and quality of the best solutions are enhanced by updating the archive residents in every iteration. Another desired characteristic of a multiobjective heuristic approach is to determine several solutions. In other words, the Pareto solutions should binge uniformly across all the objectives.

There are many multiobjective algorithms reported in the literature. The popular algorithms are Multiobjective Particle Swarm Optimization (MOPSO) (Coello 2011), Non-dominated Sorting Genetic Algorithm (NSGA) (Deb et al. 2000; Srinivas and Deb 1994), Multiobjective Ant Colony Optimization (MOALO) (Dorigo and Di Caro 1999), Multi-objective Grey Wolf Optimizer (MOGWO) (S. Mirjalili et al. 2016), Multiobjective Dragonfly Algorithm (MODA) (S. Mirjalili 2016), Multiobjective Multi-Verse Optimizer (MOMVO) (S. Mirjalili, Jangir, et al. 2017), Hybrid Multi-objective Cuckoo Search (HMOCS) (Zhang et al. 2018), Multiobjective Teaching-Learning based Algorithm (Zou et al. 2013), Multiobjective Artificial Bee Colony Algorithm (Akbari et al. 2012), Multiobjective Grasshopper Algorithm (S. Z. S. Mirjalili et al. 2018), Multiobjective Moth Flame Optimization (MOMFO) (Vikas and Nanda 2016), Multiobjective Salp Swarm Optimization (S. Mirjalili, Gandomi, et al. 2017) and Non-dominated Sorting Ions Motion Algorithm (Buch and Trivedi 2020). All these algorithms have proved their efficiency in solving the multiobjective problem. Then a question may arise: Is any new algorithm required still? According to the No Free Lunch (NFL) algorithm (Wolpert and Macready 1997), no algorithm can solve problems of all kinds. Thus, this algorithm allows proposing new algorithms or enhancement of existing ones.

The Ions Motion Algorithm (IMO) (Javidy, Hatamlou, and Mirjalili 2015) has a great exploration with fast convergence speed. The liquid and crystal phase smartly balance the exploration and exploitation stage. These features make the IMO deal with a multiobjective optimization problem potentially. Even the computational difficulty is lesser than several optimization procedures reported in the literature. Such commanding features inspired us to develop a multiobjective version of the existing single objective IMO. In this paper, the proposed algorithm is compared with other well-regarded recent optimization algorithms qualitatively and quantitatively. The remaining paper is arranged thus: Section 2 discusses the multiobjective optimization problem and associated terminology. Section 3 introduces the single objective IMO. Section 4 proposes a novel multiobjective IMO. Findings based on various performance indicators are discussed in Section 5. Section 6 concludes the work and suggests future work.

2. Introduction to Multiobjective Optimization Problem (MOOP)

In the single-objective optimization, there is a global optimum unique solution. The reason for this is the presence of only one objective in single-objective optimization problems and the existence of the most excellent unique answer. Evaluation of solutions is simple as there is only one goal and can be completed by the relational operators: ≥, >, ≤, <, or =. Such problems permit optimization issues to suitably relate the aspirant solutions and ultimately determine the finest one. While in MOOP, though, the answers should be equated with multiple criteria. Multiobjective minimization problem can be stated as follows:

$$\min F(\vec{x}) = \left\{ f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x}) \right\} \tag{1}$$

subject to

$$a_i(x) \ge 0 \quad i = 1,2,3,....,m \tag{2}$$

$$b_i(x) = 0 \quad i = 1,2,3,....,p \tag{3}$$

$$L_{ib} \le x_i \le U_{ib} \quad i = 1,2,3,...,q \tag{4}$$

Here, q presents a number of variables, $f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})$ introduces some objective functions, m and p show some inequality and equality constraints. L_{ib} and U_{ib} give lower and upper limits of the variable. Such kind of problems foil us from equating results utilizing the relational operators as there are multiple criteria to evaluate solutions. In a single objective optimization problem, a better solution can be found using a relational operator, but with various objectives, some additional operator(s) is(are) required. The primary operator to equate two solutions bearing in mind multiple objectives is called Pareto optimal dominance and is described as (Carlos A Coello Coello 2009):

The definition I: Pareto Dominance

Let us assume two vectors $\vec{a} = (a_1, a_2, ..., a_k)$ and $\vec{b} = (b_1, b_2, ..., b_k)$

Vector a dominates b if and only if:

$$\forall i \in \{1,2,...,k\}:\left[f_i(\vec{a}) \le f_i(\vec{b})\right] \wedge \exists i \in \{1,2,...,k\}:\left[f_i(\vec{a}) \prec f_i(\vec{b})\right] \tag{5}$$

By exmining Eq. (5), it may be concluded that one solution is superior to another solution if it has equal, and nonetheless, one improved value in the objectives. Under such a situation, one solution dominates another solution; otherwise, the two solutions are called Pareto optimal or non-dominated solutions. The answers to the multiobjective problem are Pareto optimal solutions. Hence, the Pareto optimality is defined as follows.

Definition II: Pareto optimality (Pareto efficiency) (Ngatchou, Zarei, and El-Sharkawi 2005)

Assuming $a \in A$, a is Pareto optimal solution if and only if:

$$\left\{\nexists \vec{b} \in A \middle| \vec{b} \prec \vec{a}\right\} \tag{6}$$

Pareto optimality or Pareto efficiency is a state of distribution of solutions from which it is not possible to achieve improvement in any one single objective or preference criterion without deterioration in a specific objective or choice criterion. Such a solution set is known as the Pareto optimal set. The prognosis of the Pareto optimal solutions in the objective search space is called Pareto optimal front.

The definition III: Pareto optimal set (Mirjalili et al., 2017)

The Pareto optimal set comprises a set of Pareto optimal solutions. Mathematically,

$$PS := \left\{\vec{a},\vec{b} \in A \middle| \nexists\ \vec{b} \prec \vec{a}\right\} \tag{7}$$

The definition IV: Pareto optimal front (Mirjalili et al., 2017)

This set consists of objective values for the solutions in the Pareto solutions set. Mathematically, it can be presented as:

$$\forall i \in \{1,2,3,...,n\}, PF := \left\{f_i(\vec{a}) \middle| \vec{a} \in PS\right\} \tag{8}$$

A quick and easy comparison of solutions of multiobjective optimization can be made with the above four equations. The group of variables, constraints and objectives create a search landscape. Considering the difficulties associated with the representation of search space for the problems with more than one objective, the researchers consider two search spaces: goal and parameter space. Likewise, in single-objective optimization, the range of variables regulates the limits of the search space in each dimension while restraints divulge them.

The overall outlines of all population-based multiobjective algorithms nearly match. They begin the optimization procedure with multiple candidate solutions. Such solutions are equated utilizing the Pareto dominance operator. In every phase of optimization, the repository/storage stores non-dominated solutions and the algorithm attempts to enhance them in the subsequent iteration(s). Different search approaches distinguish one algorithm from another to augment the non-dominated solutions.

3. Single-objective Ions Motion Algorithm (IMO)

This section first introduces the single-objective IMO algorithm. The next section presents the multiobjective form of single-objective IMO.

3.1. Ions Motion Algorithm

In 1834, Michael Faraday coined the Greek term ‘ion’. Typically, the charged particles are known as ions and can be separated into two categories: cations – ions with positive charge and anions – ions with a negative charge. Fig. 1 presents a conceptual model of force between cations and anions. The primary stimulus of the Ions Motion Algorithm is a force of attraction and repulsion between unlike and like charges, respectively. Javidy et al. (2015) proposed the population-based IMO approach stimulated by these characteristics of ions in nature.

In the IMO algorithm, anions and cations form the candidate solutions for a given optimization problem. The force of attraction/repulsion moves the ions (i.e., candidate solutions) around the search space. The ions are assessed keeping in mind the fitness value of the objective function. Anions tend to move towards the best cations while cations tend to move towards the best anions. This movement depends upon the force of attraction/repulsion between them. Such an approach guarantees improvement over iterations but does not guarantee the required exploration and exploitation of search space. Liquid and crystal are two different phases assumed to ensure necessary exploitation and exploration of search space.

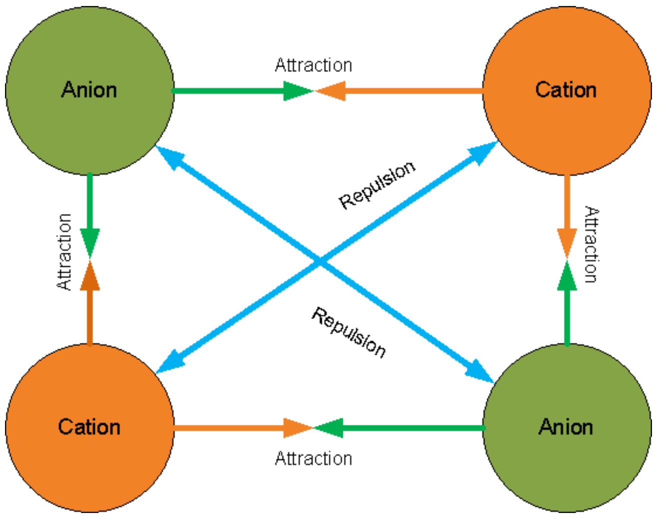


Fig. 1. Conceptual model of the force of attraction and repulsion

3.1.1. Liquid Phase

The liquid phase provides more freedom to the movement of ions, and hence, in the liquid stage, the ions can pass quickly. Also, the force of attraction is much more than the force of repulsion. Thus, the force of repulsion can be neglected to explore the search space. The distance between two ions is the only key factor considered to compute the force of attraction. So the resulting mathematical model can be proposed as:

$$Pf_{i,j} = \frac{1}{1+e^{-0.1/Pd_{i,j}}} \tag{9}$$

$$Qf_{i,j} = \frac{1}{1+e^{-0.1/Qd_{i,j}}} \tag{10}$$

Where P and Q present parameters related to anions and cations, respectively. $Pd_{i,j} = \left|P_{i,j} - Qbest_j\right|$ and $Qd_{i,j} = \left|Q_{i,j} - Pbest_j\right|$. i and j present ion index and dimension, respectively. $Pd_{i,j}$ is the distance between ith anion from the best cation in jth dimension, $Qd_{i,j}$ calculates the distance between ith the cation from the best anion in jth dimension. As presented in Eq. (9) and Eq. (10), force is inversely proportional to distances among ions. Larger the distance, lesser is the force of attraction. In other words, the force of attraction becomes less when the distance grows higher from the best ion with the opposite charge.

According to Eq. (9) and Eq. (10), the value of force varies between 0.5 to 1. $Pf_{i,j}$ and $Qf_{i,j}$ are the resultant attraction forces of anions and cations, respectively. After force calculation, the position of positive and negative ions is updated as per the following equations:

$$P_{i,j} = P_{i,j} + Pf_{i,j}\left(Qbest_j - P_{i,j}\right) \tag{11}$$

$$Q_{i,j} = Q_{i,j} + Qf_{i,j}\left(Pbest_j - Q_{i,j}\right) \tag{12}$$

$Pf_{i,j}$ and $Qf_{i,j}$ are the resulting attraction forces between opposite ions while $Qbest_j$ and $Pbest_j$ present the best cations and anions, respectively. The attraction force between ions guarantees exploration. Referring to Eqs. (9-12), the conclusion can be drawn that in the liquid phase, there is no involvement of the random component. Fig. 2 presents an abstract model of the movement of ions in the liquid stage. With an increasing number of iterations, more and more ions start interacting, converging towards the best ion with an opposite charge, and hence, exploration gradually decreases. This phenomenon is precisely like a conversion from liquid to crystal state observed in nature. The search agents, i.e., ions, also enter crystal state, finally converging towards the best solution in search space.

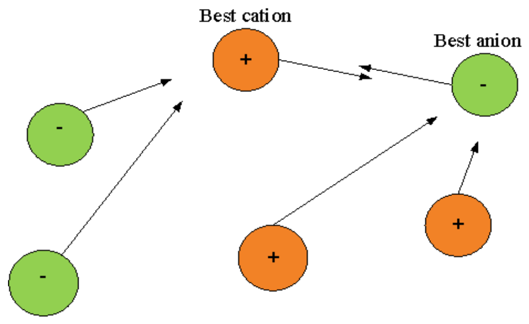


Fig. 2. Ions movement towards the best ions in the liquid phase

3.1.2. Crystal Phase

In this stage, the ions congregate to the optimal solution. Convergence has already taken place. Since the search space has an unknown form, occasionally convergence gets trapped into local minima. A separate mechanism is proposed at the crystal stage to avoid trapping of solutions in local minima. The cations and anions in the crystal phase are organized to maximize their force of attraction. When an outside force is applied to the same charges in the crystal phase, the resultant repulsion force cracks the crystal apart. Mathematically, the mechanism to overcome local optimum trapping can be demonstrated as below:

```
if (QbestFit ≥ QworstFit/2 and PbestFit ≥ PworstFit/2)
        if rand () > 0.5
            $P_i = P_i + \varphi_1 \times (Qbest - 1)$
        else
            $P_i = P_i + \varphi_1 \times (Qbest)$
        end if
        if rand () > 0.5
            $Q_i = Q_i + \varphi_2 \times (Pbest - 1)$                (13)
        else
            $Q_i = Q_i + \varphi_2 \times (Pbest)$
        end if
        if rand () < 0.5
            Re-initialize $P_i$ and $Q_i$
        end if
end if
```

Where, φ_1 and φ_2 are random numbers between $[-1,1]$ and $rand(\ )$ is a random number between $[0,1]$. $QbestFit$ and $QworstFit$ present fitness of the best and worst cations. $PbestFit$ and $PworstFit$ are the fitness of best and worst anions. The best fitness of anions and cations should be better than or equal to the average competence of the worst anions and cations. If this situation is met, ions are arbitrarily navigated in search space to circumvent stagnation adjacent to local minima. Again, ions enter the liquid state until termination criteria are met.

It should be noticed here that in Eq. (13), four, instead of two, conditions are proposed to achieve different behavior of the proposed algorithm. These four conditions are presented below:

1. Both first if-else statements are met:
$P_i = P_i + \varphi_1 \times (Qbest - 1)$

$Q_i = Q_i + \varphi_2 \times (Pbest - 1)$

2. Only the first if-else statement is met:
$P_i = P_i + \varphi_1 \times (Qbest - 1)$
$Q_i = Q_i + \varphi_2 \times (Pbest)$

3. Only the second if-else statement is met:
$P_i = P_i + \varphi_1 \times (Qbest)$
$Q_i = Q_i + \varphi_2 \times (Pbest - 1)$

4. Both conditions are not met:
$P_i = P_i + \varphi_1 \times (Qbest)$
$Q_i = Q_i + \varphi_2 \times (Pbest)$

In contrast, merging the first two if-else statements will result in only two conceivable combinations:

5. Collective if-else sentences are fulfilled:
$P_i = P_i + \varphi_1 \times (Qbest - 1)$
$Q_i = Q_i + \varphi_2 \times (Pbest - 1)$

6. Combined if-else sentences are not fulfilled:
$P_i = P_i + \varphi_1 \times (Qbest)$
$Q_i = Q_i + \varphi_2 \times (Pbest)$

Thus, splitting two conditions into four provide different behavior for the IMO which helps avoid local optimal entrapment. Fig. 3 presents the standard steps of the Ions Motion Algorithm. The IMO starts with a random group of solutions. The arbitrary collection of solutions during initialization is generated using $r(ub_i - lb_i) + lb_i$ where r is a random number with uniform distribution in the interval $(0,1)$. ub_i and lb_i represent upper and lower bound respectively of ith variable. At this phase, ions are equally separated into a set of anions and cations, respectively. The fitness of each anion and cation is calculated, and according to fitness, the best and worst anions/cations are selected and saved. The attraction forces and positions are updated using Eqs. (9-12). During each iteration, if the condition of the crystal phase is met, the ions go into the crystal phase. Till the satisfaction of termination criteria, ions keep going between solid and liquid phases. In the end, the best ion is reported as the best approximation of the global solution.

4. Multiobjective Ions Motion Algorithm (MOIMO)

The single objective IMO algorithm can drive ions towards the best ion and update it throughout the iterations. But, this approach cannot solve multiobjective problems mainly owing to the following two reasons:

- IMO cannot store multiple best solutions as it preserves a unique solution as the optimum solution.
- IMO updates the ion position concerning the unique best solution attained up to now during each iteration, but MOOPs have no single best solution.

The first limitation is handled by incorporating the IMO algorithm with the storage of ions. This storage preserves the best non-dominated solutions attained so far throughout optimization and is very close to the archive (storage) mechanism employed in MOPSO (Coello & Lechuga 2002) and Pareto Archived Evolution Strategy (PAES) (Knowles and Corne 1999). The storage has the maximum storage capacity to stock the limited quantity of solutions. Throughout optimization, every ion is compared with every storage element using Pareto dominance operators. Such a comparison may provide us following different cases:

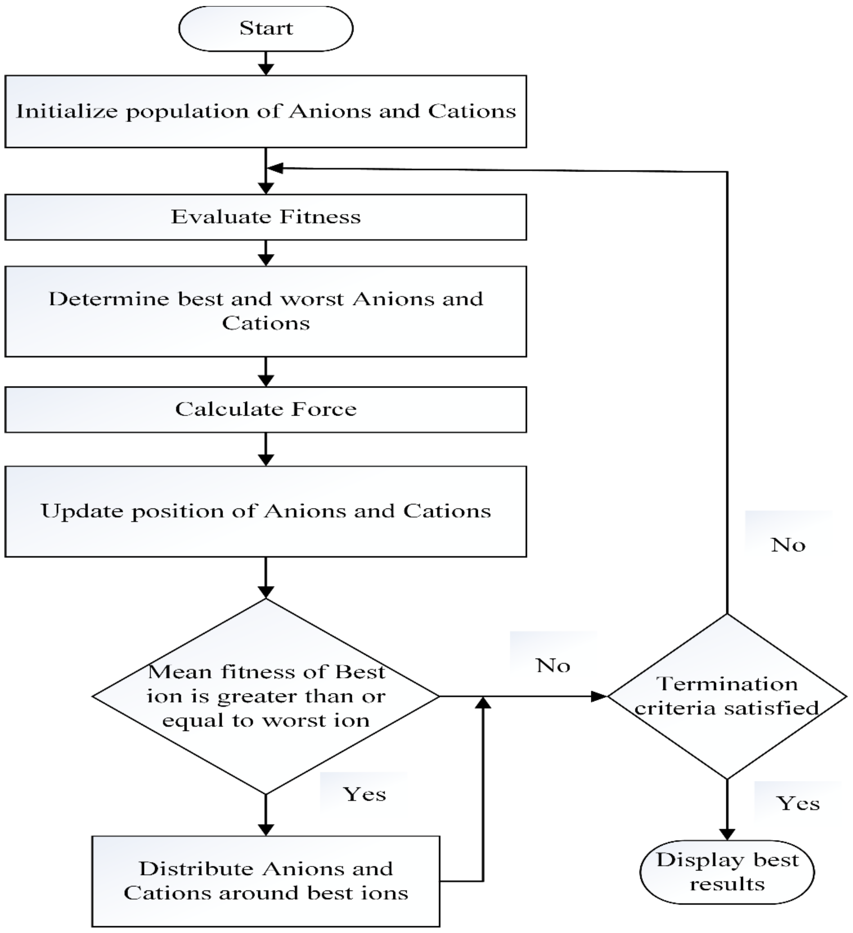

Start
Initialize population of Anions and Cations
Evaluate Fitness
Determine best and worst Anions and Cations
Calculate Force
Update position of Anions and Cations
Mean fitness of Best ion is greater than or equal to worst ion
No
Yes
Distribute Anions and Cations around best ions
Termination criteria satisfied
No
Yes
Display best results

Fig. 3. The standard procedure of the IMO algorithm

1. If the new solution dominates archive residents, then they should be swapped.
2. If the new solution leads a group of solutions in the storage, they all should be deleted from the storage, and a new solution should be added in the storage.
3. If the storage is full and the new solution is non-dominated in comparison with the storage elements, then the most straightforward means is to remove any solution randomly and swap it with the new non-dominated solution. However, as the apriori method should be able to determine uniformly dispersed Pareto front, the best candidate to be eliminated from the storage is the one in the densest area. Such a technique will improve the distribution of repository residents throughout the iterations.

The number of neighboring solutions with a specified maximum distance is counted and assumed to find the non-dominated solution(s) with a populated neighborhood. This distance is calculated by $d = \frac{max-min}{repository\ size}$ where max and min are two vectors for storing maximum and minimum objective values, respectively. The storage with one solution in each segment is an ideal case. Based on the number of neighboring repository residents, each solution is assigned a rank. After assigning a rank to every storage element depending upon the number of neighboring solutions, a roulette wheel mechanism is employed to choose one of them. Higher the rank of a solution, higher the likelihood of eliminating it from the storage.

The second component is a leader selection strategy that assists in choosing the best ions as there are multiple best solutions in a multiobjective search space. Again, the ion can be selected arbitrarily from the storage. However, a more suitable method is to choose an answer from the least populated region. A similar mechanism is applied using the roulette wheel selection and the same ranking process. The main difference between archive maintenance and leader selection is that in archive maintenance, the solution is likely to be chosen from the crowded neighborhood while in contrast, for leader selection strategy, the solution is expected to be selected from a less crowded area. Fig. 4 presents a schematic representation of an archive-based plan for a multiobjective optimization approach.

In a nutshell, the MOIMO approach first generates the population of ions concerning the upper and lower bounds of variables. This approach then computes the objective values of each ion and determines the non-dominated ones. If the storage has a vacancy, then the non-dominated solutions are added into it. If the storage is completely occupied, the archive maintenance is run to remove the solution with a crowded neighborhood. In this case, the solutions are ranked and chosen through a roulette wheel. After eliminating the required number of solutions from the storage, the new solutions are added to the storage. After updating the storage, an ion is selected from the non-dominated solutions having the least crowded neighborhood. The next step is to update the positions. All the above steps are repeated (except initialization), till the termination criteria are met.

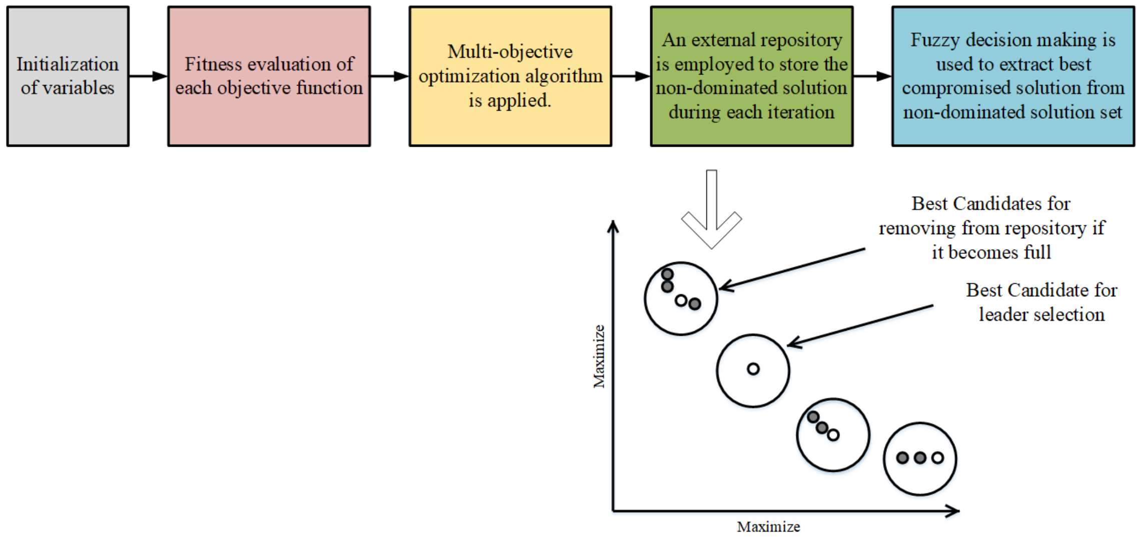

Initialization of variables
Fitness evaluation of each objective function
Multi-objective optimization algorithm is applied.
An external repository is employed to store the non-dominated solution during each iteration
Fuzzy decision making is used to extract best compromised solution from non-dominated solution set
Best Candidates for removing from repository if it becomes full
Best Candidate for leader selection
Maximize
Maximize

Fig. 4. Schematic representation of archive-based multiobjective optimization algorithm

The following vital observations can be made regarding the MOIMO algorithm:

1. The non-dominated solutions attained are deposited in storage, so they never get lost even if the whole population worsens during an iteration.
2. The solutions with the dense neighborhood are rejected whenever storage maintenance is called. This results in refining the diversity of non-dominated solutions across all objectives.
3. The best anion and cation are selected from the list of repository members having the smallest number of neighboring solutions which concentrates the search to the less dense regions of the Pareto front and thus enhances the diversification.
4. MOIMO inherits the search mechanism from basic IMO.

The computational complexity of MOIMO is $O(PQ^2)$, where P and Q are numbers of objectives and solutions, respectively. Thus, computational complexity is comparable to PAES (Knowles and Corne 1999) and MOPSO (C A Coello Coello and Lechuga 2002). With the procedure mentioned above, MOIMO can find Pareto optimal solutions, store them in the repository and enhance their dispersal. The next section discusses the performance of MOIMO on the standard benchmark functions.

5. Findings on Test Functions

5.1. Experimental Setup

Various benchmark test functions having diverse characteristics are employed to scrutinize the performance of multiobjective optimization algorithms as different test functions can challenge an algorithm from different standpoints. The benchmark functions used are (Mirjalili et al., 2016):

- Unconstrained multiobjective test functions (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, SCH1, SCH2)
- Constrained multiobjective test functions (BEL, BNH, CONSTR, CF1, OSY)
- Engineering design multiobjective problems (3 Bar Truss, Brushless DC Wheel Motor, Pressure Vessel, Satellite Pipe, Welded Beam, Car Crash Design Problem)

The performance of MOIMO is compared with other well-regarded algorithms in literature like MOMVO (S. Mirjalili, Jangir, et al. 2017), MODA (S. Mirjalili 2016), and MOALO (S. Mirjalili, Jangir, and Saremi 2017). The results are evaluated quantitatively and qualitatively. For qualitative comparison, the best Pareto optimal front over ten independent runs is chosen. However, the qualitative comparison does not indicate how much an algorithm is better than the other one.

And hence, different quantitative performance indicators are employed. The first metric is the Generational Distance (GD) (Van Veldhuizen 1999). GD is calculated between the true Pareto front and obtained the Pareto front. It calculates the average distance between the obtained Pareto front and the true Pareto front and can be mathematically presented as:

$$GD = \frac{\left(\sum_{p=1}^{k}(d_i)\right)}{k} \tag{14}$$

where,

$$d_i = \left(\sum_{i=1}^{n}\left(PF_{i.p}^{o} - PF_{i.p}^{t}\right)^2\right)^{1/2} \tag{15}$$

Here k is the number of Pareto solutions, n is the number of objective functions, $PF_{i.p}^{o}$ indicate the p^{th} obtained Pareto solution for the *ith* objective, and $PF_{i.p}^{t}$ indicate the nearest point on the true Pareto front from $PF_{i.p}^{o}$.

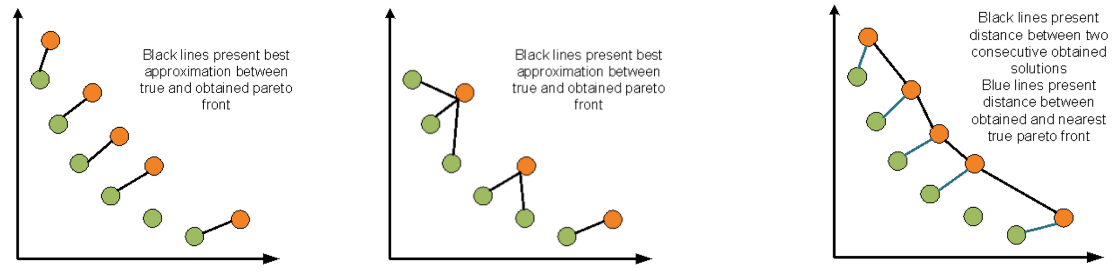

Black lines present best approximation between true and obtained pareto front
Black lines present best approximation between true and obtained pareto front
Black lines present distance between two consecutive obtained solutions
Blue lines present distance between obtained and nearest true pareto front

Fig. 5. Schematic representation of (a) Generational Distance (GD) (b) Inverse Generational Distance (IGD) and (c) spread

Another indicator used is Inverse Generational Distance (IGD) (Coello and Cortés 2005). It calculates the distance between each element of the obtained front from the true Pareto front keeping true Pareto front as a reference. It is mathematically defined as:

$$IGD = \frac{\sqrt{\sum_{i=1}^{Q} d_i^2}}{Q} \tag{16}$$

where Q presents the number of solutions in the true Pareto front and d_i is the Euclidean distance between each of the solutions in true Pareto front and the nearest member from the set of non-dominated solutions found by the algorithm. The IGD measures both the diversity and the convergence of an obtained non-dominated solution set. The smaller value of IGD presents closeness between true and obtained Pareto front.

Spread (Deb, 2001) and spacing (S) (Schott, 1995) help measure coverage. Coverage helps identify the distribution of obtained solutions over true Pareto front. Mathematically, the spread is defined as:

$$\Delta = \frac{\sum_{j=1}^{n} d_j^{ex} + \sum_{p=2}^{k}\left|d_p - \overline{d}\right|}{\sum_{j=1}^{n} d_j^{ex} + (k-1)\overline{d}} \tag{17}$$

Here, d_p is the Euclidian distance between two consecutive points on the obtained Pareto front and d_j^{ex} is the Euclidian distance between the obtained Pareto front and true Pareto front. $\overline{d}$ is the average of d_p. Spread verifies the condition for the obtained Pareto front to cover the true Pareto front. Smaller the value of spread, better the spread, i.e., the uniform distribution of obtained solutions. Another indicator, i.e., the spacing is defined as:

$$S = \frac{1}{k-1}\sum_{p=2}^{k}\left(D_p - \overline{D}\right)^2 \tag{18}$$

Here, D_p is the absolute difference between two consecutive solutions in the obtained Pareto front. It is defined as:

$$D_p = \sum_{i}^{n}\left|PF^0_{i.(p-1)} - PF^0_{i.p}\right| \tag{19}$$

where, $\overline{D}$ is the average of all D_p. Spacing specifies the spread of the obtained Pareto front. The higher value of S indicates the better coverage of the obtained Pareto solutions. Fig. 5. schematically represents the generational distance, inverse generational distance and spread. Each algorithm is run for 10 independent times for 200 iterations each. The archive size and the number of search agents are set to 100 for constrained and unconstrained functions while they are set to 200 for real-world benchmark functions. For a fair comparison, the population size is set to 100 for constrained and unconstrained benchmark functions. For real-world engineering design problems, the population size is set to 200. The results are tabulated in the next subsections and the best values are highlighted in bold.

5.2. Results on Unconstrained Test Functions

The first set of test problems contain unconstrained standard benchmark functions. Seven different benchmark functions, i.e., ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, SCHN1 and SCHN2 are employed to check the performance of the MOIMO. Fig**. 6.** and Table 1 to Table 4 present qualitative and quantitative performance assessment for the unconstrained test functions. As shown in Fig**. 6.** , for all the unconstrained test functions, MOIMO provides better coverage as compared to the rest of the algorithms. In one of the most challenging test functions, i.e., ZDT3, MOIMO provides uniform coverage in all the algorithms.

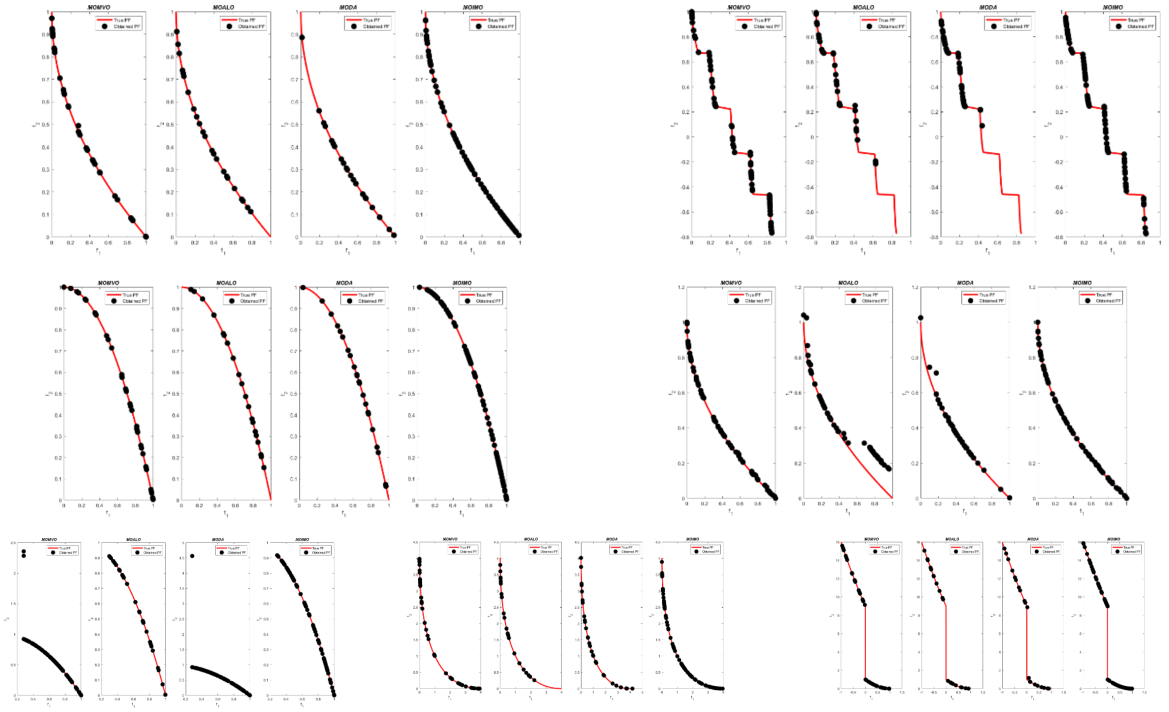


Fig. 6. Best Pareto front for unconstrained benchmark functions

While analyzing the results quantitatively, it is seen that, for most performance indicators, the MOIMO provides significantly better results than the rest of the algorithms. MOIMO is compared for the average value of each performance indicator and standard deviation obtained over 10 independent runs. These results display the superiority of MOIMO showing higher accuracy and better robustness. Thus, MOIMO has the potential to outclass the rest of the algorithms in achieving Pareto optimal front with non-convex non-uniform regions.

Table 1
Results of the multiobjective algorithm (using GD) on the unconstrained test functions

	MOMVO	MOALO	MODA	MOIMO
ZDT1				
Avg. GD	0.0065457	0.00010694	**9.4009e-05**	0.00015758
SD of GD	0.013886	**2.207e-05**	3.6339e-05	7.1257e-05
ZDT2				
Avg. GD	0.017519	**8.2369e-05**	0.0018219	9.6513e-05
SD of GD	0.038833	3.3481e-05	0.0038699	**8.5783e-06**
ZDT3				
Avg. GD	0.00045131	0.00059499	0.00051111	**0.00031746**
SD of GD	4.0207e-05	0.00016385	0.0003006	**6.1964e-06**
ZDT4				
Avg. GD	0.0004134	0.0075542	0.0029169	**0.00031644**
SD of GD	0.00012338	0.006953	0.0041258	**8.5969e-05**
ZDT6				
Avg. GD	0.016918	0.073206	0.08976	**8.7355e-05**
SD of GD	0.011282	0.087903	0.14623	**4.2786e-06**
SCH1				
SD of GD	1.913e-05	3.2288e-05	0.00012406	**1.0351e-05**
Avg. GD	0.00024794	0.00024983	0.00033026	**0.00023967**
SCH2				
SD of GD	2.5767e-05	**2.301e-05**	0.00015192	0.0025352
Avg. GD	**1.786e-06**	2.6257e-06	0.00015659	0.00073378

Table 2
Results of the multiobjective algorithm (using IGD) on the unconstrained test functions

	MOMVO	MOALO	MODA	MOIMO
ZDT1				
Avg. IGD	0.0065457	**0.00010694**	0.0018231	0.00054853
SD of IGD	0.013886	**2.207e-05**	0.00055216	9.422e-05
ZDT2				
Avg. IGD	0.0014645	0.0019369	0.0021012	**0.00064484**
SD of IGD	0.00013103	0.00061345	0.0015287	**8.5653e-05**
ZDT3				
Avg. IGD	**0.00045131**	0.00059499	0.00051111	0.00068141
SD of IGD	**4.0207e-05**	0.00016385	0.0003006	0.0001457
ZDT4				
Avg. IGD	0.0022795	0.0030114	0.0028121	**0.00053503**
SD of IGD	0.0032244	0.00089163	0.0017037	**5.5694e-05**
ZDT6				
Avg. IGD	**0.00054492**	0.0017166	0.0086791	0.00057224
SD of IGD	0.00010866	0.0010588	0.016899	**4.4493e-05**
SCH1				
SD of IGD	0.002219	0.003891	0.0018651	**0.0010189**
Avg. IGD	0.0011036	0.0030438	0.00030546	**0.0002428**
SCH2				
SD of IGD	0.00054056	0.00080233	0.0007836	**0.00025411**
Avg. IGD	7.0394e-05	0.00015486	0.00012158	**3.2892e-05**

Table 3
Results of the multiobjective algorithm (using spread) on the unconstrained test functions

	MOMVO	MOALO	MODA	MOIMO
ZDT1				
Avg. Spread	1.4095	1.4769	1.4839	**0.89417**
SD of Spread	0.038734	**0.038021**	0.1671	0.1075
ZDT2				
Avg. Spread	1.4492	1.4704	1.4719	**0.85165**
SD of Spread	0.049026	**0.026366**	0.083441	0.035557
ZDT3				
Avg. Spread	1.1088	1.363	1.3287	**1.0253**
SD of Spread	0.055513	**0.043786**	0.17185	0.076027
ZDT4				
Avg. Spread	0.88654	1.2407	1.3232	**0.84776**
SD of Spread	0.058543	0.074618	0.10319	**0.056613**
ZDT6				
Avg. Spread	1.075	1.3359	1.3769	**0.90682**
SD of Spread	0.07035	0.091983	0.17152	**0.038463**
SCH1				
SD of Spread	1.2704	1.5001	1.4519	**0.87412**
Avg. Spread	0.065065	0.059148	0.046144	**0.044177**
SCH2				
SD of Spread	1.3494	1.6203	1.5984	**1.001**
Avg. Spread	0.070502	0.074842	0.0639	**0.028742**

Table 4
Results of the multiobjective algorithm (using spacing) on the unconstrained test functions

	MOMVO	MOALO	MODA	MOIMO
ZDT1				
Avg. Spacing	**0.088253**	0.012002	0.013515	0.057615
SD of Spacing	0.12858	0.0086463	**0.0059354**	0.0083665
ZDT2				
Avg. Spacing	**0.18524**	0.016203	0.035237	0.055014
SD of Spacing	0.38283	**0.0042029**	0.046016	0.014261
ZDT3				
Avg. Spacing	0.12386	0.03244	0.049203	**0.13544**
SD of Spacing	0.030695	**0.015926**	0.03728	0.027096
ZDT4				
Avg. Spacing	0.044365	0.044564	0.040731	**0.052348**
SD of Spacing	**0.0044424**	0.046192	0.039918	0.010145
ZDT6				
Avg. Spacing	0.19458	0.15945	**0.23784**	0.071907
SD of Spacing	0.096783	0.10851	0.22936	**0.0080407**
SCH1				
Avg. of Spacing	0.19661	0.082656	0.19319	**0.46286**
SD of Spacing	**0.096635**	0.041882	0.068511	0.067796
SCH2				
Avg. of Spacing	1.472	0.071143	0.29546	**2.0499**
SD of Spacing	0.41896	**0.094253**	0.39373	0.87483

5.3. Results on Constrained Test Functions

The next group of test function comprises five constrained benchmark functions. We must include the constraint handling method with MOIMO to make it capable of solving such problems. Identifying an appropriate constraint handling method is out of the scope. In this work, a death penalty function (Carlos, 2000) is used to punish search agents that violate any of the constraints at any stage. For equating algorithms, four metrics are applied in this research: GD, IGD, metric of spread and metric of space. Table 5 to Table 8 present quantitative results. These performance pointers permit us to enumerate and equate algorithms regarding diversification and intensification. Fig. 3 presents the shape of the best Pareto front achieved by the four algorithms on constrained benchmark functions. Reviewing these figures, MODA presents poor performance, notwithstanding its good coverage in a few cases. However, MOIMO and MOMVO both offer a better convergence close to all true Pareto fronts.

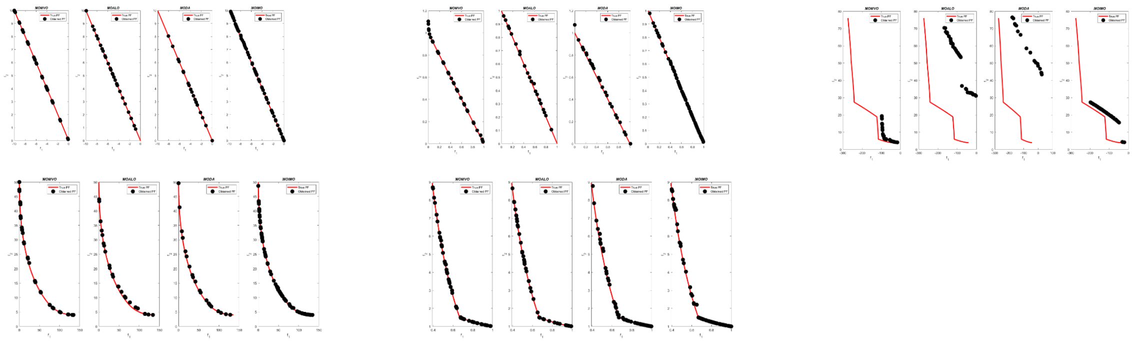


Fig. 3. Best pareto front for constrained benchmark functions

Table 5
Results of the multiobjective algorithm (using GD) on the constrained test functions

	MOMVO	MOALO	MODA	MOIMO
BEL				
Avg. GD	**7.4819e-05**	0.00022544	0.0001356	0.00023102
SD of GD	**4.8942e-05**	0.00026187	0.0001369	0.00026533
BNH				
Avg. GD	0.0022643	0.00094086	0.00061468	**0.00039748**
SD of GD	0.00032499	0.00030889	0.0003078	**3.5912e-05**
CF1				
Avg. GD	0.005943	0.002448	0.002289	**0.0020175**
SD of GD	0.0010235	0.0001741	0.0002851	**7.7036e-05**
CONSTR				
Avg. GD	**0.00033082**	0.00087881	0.0012952	0.00072863
SD of GD	**4.0943e-05**	0.00062172	0.00058203	0.00013165
OSY				
Avg. GD	**0.0078333**	0.027875	0.04946	0.011565
SD of GD	**0.00093499**	0.018849	0.012052	0.0023181

Table 6

Results of the multiobjective algorithm (using IGD) on the constrained test functions

	MOMVO	MOALO	MODA	MOIMO
BEL				
Avg. IGD	0.0011629	0.0012543	0.0018785	**0.00039751**
SD of IGD	0.0001322	0.00030576	0.00082551	**2.812e-05**
BNH				
Avg. IGD	0.0022643	0.0030569	0.0032425	**0.0010153**
SD of IGD	0.00032499	0.00073384	0.0011784	**0.00010699**
CF1				
Avg. IGD	0.005943	0.0084038	0.0069698	**0.0026365**
SD of IGD	0.0010235	0.00089992	0.001894	**0.00061807**
CONSTR				
Avg. IGD	0.00087387	0.0010705	0.0011423	**0.00081589**
SD of IGD	0.00014332	**0.00014314**	0.00040386	0.00026318
OSY				
Avg. IGD	0.010584	0.0095221	0.011433	**0.0070975**
SD of IGD	0.0044182	0.0016712	0.0029412	**0.0016022**

Table 7
Results of the multiobjective algorithm (using spread) on the constrained test functions

	MOMVO	MOALO	MODA	MOIMO
BEL				
Avg. Spread	1.5036	1.4448	1.5583	**0.92292**
SD of Spread	**0.023962**	0.058427	0.095943	0.063347
BNH				
Avg. Spread	1.4088	1.5122	1.4977	**0.84389**
SD of Spread	0.074574	**0.014647**	0.08599	0.062455
CF1				
Avg. Spread	1.3621	1.517	1.4734	**0.78473**
SD of Spread	**0.022836**	0.040547	0.14169	0.065962
CONSTR				
Avg. Spread	1.1651	1.2998	1.1346	**0.9405**
SD of Spread	**0.048443**	0.071313	0.06642	0.064704
OSY				
Avg. Spread	0.93818	1.2396	1.2547	**0.92908**
SD of Spread	0.037576	0.10688	**0.027483**	0.034359

Table 8
Results of the multiobjective algorithm (using spacing) on the constrained test functions

	MOMVO	MOALO	MODA	MOIMO
BEL				
Avg. Spacing	0.0039105	0.0051075	0.010523	**0.012261**
SD of Spacing	**0.0033895**	0.0043164	0.021808	0.016845
BNH				
Avg. Spacing	8.482	3.9153	4.2928	**19.4547**
SD of Spacing	5.4558	2.3859	2.8649	**2.3896**
CF1				
Avg. Spacing	**0.014848**	0.0037717	0.0035515	0.0046881
SD of Spacing	**0.024059**	0.0026615	0.0021676	0.0018632
CONSTR				
Avg. Spacing	0.78491	0.46369	0.76356	**0.95727**
SD of Spacing	0.2387	0.21466	0.30439	**0.46124**
OSY				
Avg. Spacing	22.9733	5.2117	9.9368	**31.2487**
SD of Spacing	5.8054	**3.2005**	5.6156	5.9582

5.4. Engineering Benchmark Design Problems

The final set of benchmark functions is the most stimulating one comprising six real-world engineering design problems. These problems have varied constrained characteristics. Table 9 to Table 12 present a quantitative comparison of all the four algorithms for different performance indicators. The results in these tables are in line with those in the preceding sections, in which the MOIMO approach generally presented superior convergence and coverage. Because of the difficulty of these benchmark problems, the better results exceedingly sustenance the dominance of the MOIMO approach and its applicability.

Fig. 4 presents the best Pareto optimal fronts for real-world engineering design problems. It may be seen in this figure that MOIMO has better qualitative characteristics amongst all algorithms despite multimodal search space and the existence of various limitations.

Table 9

Outcomes of the multiobjective algorithm (GD) on the engineering design functions

	MOMVO	MOALO	MODA	MOIMO
Design of 3-Bar Truss				
Avg. GD	0.00081641	0.0014405	0.0011318	**0.00062503**
SD of GD	0.00055249	0.0010217	0.00050327	**0.0004354**
Brushless DC Machine Design Problem				
Avg. GD	**0.0013737**	0.0018788	0.0043479	0.0027073
SD of GD	0.0013662	**0.0005236**	0.0011936	0.00062814
Pressure Vessel Design				
Avg. GD	0.00154	0.0088965	0.0036704	**0.0011863**
SD of GD	**0.00016098**	0.004002	0.00096662	0.0001821
Satellite Pipe Design				
Avg. GD	**0.00060649**	0.0010924	0.0032735	0.00063076
SD of GD	0.00030408	0.00054764	0.0028936	**0.00016805**
Welded Beam Design Problem				
Avg. GD	0.0049408	0.0083299	**0.0018358**	0.0030226
SD of GD	0.0039537	0.0062062	**0.00097582**	0.00098187
Car Crash Design Problem				
Avg. GD	**0.0041605**	0.076208	0.074507	0.059254
SD of GD	**0.0015295**	0.033869	0.0065088	0.018628

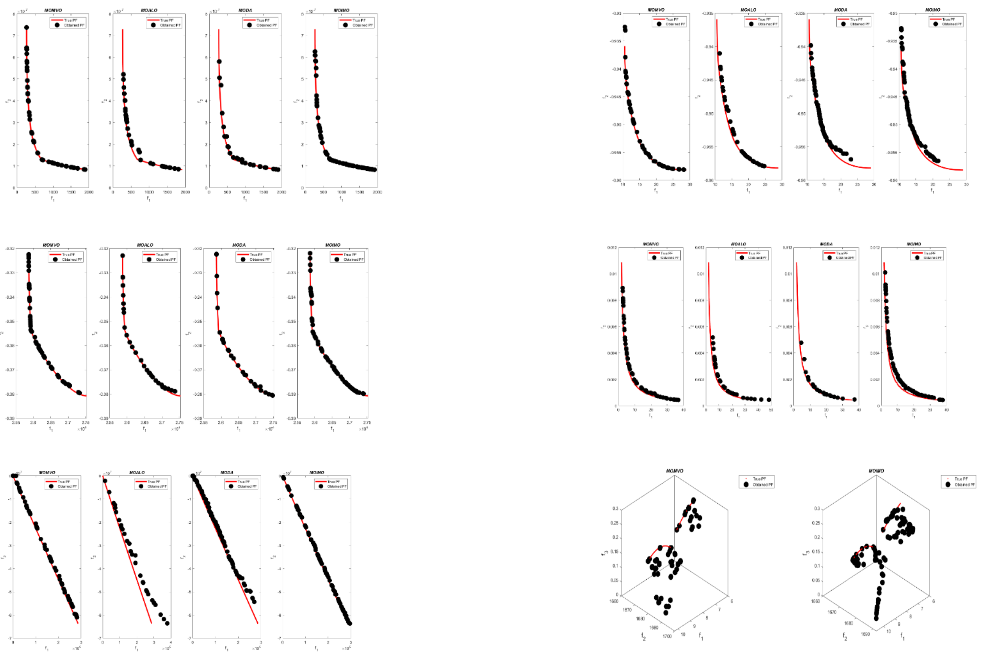


Fig. 4. Best Pareto front for real-world engineering benchmark functions

Table 10
Outcomes of the multiobjective algorithm (IGD) on the engineering design functions

	MOMVO	MOALO	MODA	MOIMO
3 Bar Truss Design Problem				
Avg. IGD	0.00081641	0.0014405	0.0011318	**0.00062503**
SD of IGD	0.00055249	0.0010217	0.00050327	**0.0004354**
Brushless DC Machine Design Problem				
Avg. IGD	**0.0013737**	0.0018788	0.0043479	0.0027073
SD of IGD	0.0013662	**0.0005236**	0.0011936	0.00062814
Pressure Vessel Design				
Avg. IGD	0.00047789	0.0014181	0.00074622	**0.00037128**
SD of IGD	**3.914e-05**	0.00043574	0.00016207	6.4527e-05
Satellite Pipe Design				
Avg. IGD	**0.00048671**	0.00072714	0.00083374	0.00067951
SD of IGD	0.00015372	**0.00014836**	0.00015485	0.00046841
Welded Beam Design Problem				
Avg. IGD	0.0055088	0.010773	0.014105	**0.0044458**
SD of IGD	0.003133	0.0039621	0.0062055	**0.0030843**
Car Crash Design Problem				
Avg. IGD	0.0041605	0.0080252	0.005071	**0.0021731**
SD of IGD	0.0015295	0.0059468	0.0014011	**0.00064515**

Table 11

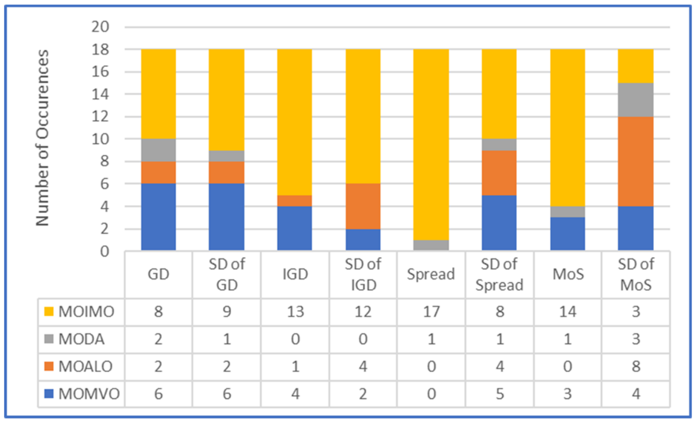
Outcomes of the multiobjective algorithm (Spread) on the engineering design functions

	MOMVO	MOALO	MODA	MOIMO
3 Bar Truss Design Problem				
Avg. Spread	1.2768	1.3753	1.4348	**0.99958**
SD of Spread	0.063482	0.06986	0.081976	**0.037465**
Brushless DC Machine Design Problem				
Avg. Spread	1.0505	1.1656	0.89866	**0.88668**
SD of Spread	0.069106	0.096319	0.087243	**0.065443**
Pressure Vessel Design				
Avg. Spread	0.99371	1.3947	**0.73348**	0.7553
SD of Spread	**0.032797**	0.057863	0.063679	0.05195
Satellite Pipe Design				
Avg. Spread	1.0261	1.2965	1.4348	**0.90803**
SD of Spread	0.055868	0.10855	0.068078	**0.034882**
Welded Beam Design Problem				
Avg. Spread	1.0644	1.322	1.2264	**0.78972**
SD of Spread	0.10156	0.047291	0.22633	**0.019262**
Car Crash Design Problem				
Avg. Spread	1.0185	1.2545	1.1225	**0.79357**
SD of Spread	**0.073509**	0.090501	0.13589	0.093567

Table 12
Outcomes of the multiobjective algorithm (Spacing) on the engineering design functions

	MOMVO	MOALO	MODA	MOIMO
3 Bar Truss Design Problem				
Avg. Spacing	161.0758	48.0183	134.1771	**409.7276**
SD of Spacing	40.4036	**24.9562**	62.3906	104.2308
Brushless DC Machine Design Problem				
Avg. Spacing	1.5504	0.58266	1.0003	**1.545**
SD of Spacing	0.3786	**0.19288**	0.74956	0.36674
Pressure Vessel Design				
Avg. Spacing	8346712.3552	2309040.2674	8599005.8531	**9895393.2942**
SD of Spacing	2829417.1318	**385083.5572**	2754131.991	1831215.4646
Satellite Pipe Design				
Avg. Spacing	209.9238	109.1347	87.9714	**270.4866**
SD of Spacing	71.5969	64.8868	**57.9485**	75.78
Welded Beam Design Problem				
Avg. Spacing	5.7868	2.7399	2.0075	**6.5448**
SD of Spacing	1.6432	**1.4783**	2.1302	2.7005
Car Crash Design Problem				
Avg. Spacing	2.9978	1.4769	4.0722	**5.5785**
SD of Spacing	0.61387	0.92514	**0.40501**	1.0367

Fig. 5 presents the overall performance comparison of all the algorithms in terms of best values of performance indicators and its (their) standard deviation over 10 independent runs. MOIMO obtains the best value of average GD and IGD for 8 and 13 times, respectively. GD and IGD are indicators of proximity to the true Pareto front. For 17 instances, MOIMO obtains the best value of spread presenting better coverage and convergence in all algorithms. MoS reflects the coverage of solutions. For 14 cases, MOIMO gets the best values of MoS. Except for MoS, for all three performance indicators, MOIMO presents minimum standard deviation proving the consistency of results. Thus, we can conclude that MOIMO provides the best results across all the algorithms for the majority of performance indicators.



	GD	SD of GD	IGD	SD of IGD	Spread	SD of Spread	MoS	SD of MoS
🟨 MOIMO	8	9	13	12	17	8	14	3
⬜ MODA	2	1	0	0	1	1	1	3
🟧 MOALO	2	2	1	4	0	4	0	8
🟦 MOMVO	6	6	4	2	0	5	3	4

Fig. 5. Overall performance comparison of all the algorithms for benchmark test functions

To sum up, the outcomes demonstrate that MOIMO is highly capable and modest compared to the existing well-regarded approaches. High coverage and convergence can be two key reasons behind it. Higher convergence is the result of leader selection strategy wherein candidate solutions update their position considering the best non-dominated solution. High coverage is the result of storage maintenance and leader selection strategy. Since the leader is selected from the least populated region and solutions are removed from the dense area, the MOIMO has better coverage and diversity in all objectives. Despite all these merits, the present version of MOIMO can be applied to maximum four objectives. With an increasing number of objectives, MOIMO becomes less effective. This is because, for problems with more than four objectives, the archive gets full very quickly on account of a considerable number of non-dominated solutions. Besides, this approach is appropriate only for an objective function with a continuous variable and requires necessary changes for application on problems with discrete variables. The findings evidenced that MOIMO can be very useful for optimizing multiobjective problems. The MOIMO approach presented better coverage (diversification) and convergence (intensification). The superior intensification of MOIMO is owing to the updating solutions about the best non-dominated solutions attained up to now. The solutions move towards the best solutions. Also, the high convergence initiates from the adaptive mechanism, which quickens the travels of ions to the best non-dominated solutions obtained up to now in the storage. The storage maintenance and leader selection strategy result in higher coverage. It is worth stating here that as the updating mechanism of the ions in MOIMO is like IMO, MOIMO receives good exploration, local solutions evasion, fast convergence, and exploitation from this algorithm. Apart from convergence and coverage, algorithms are compared in terms of computational time in Table 13. For five different instances, MOIMO takes minimum simulation time, while for 12 examples, it stands second closely following the MOMVO. Thus, in terms of computational time, MOIMO presents excellent performance.

Table 13
Comparison of algorithms in terms of computational time

Sr. No.	Benchmark test function	Algorithms			
		MOMVO	MOALO	MODA	MOIMO
1	ZDT1	**12.6218**	15.350	63.934	16.993
2	ZDT2	**1.889**	2.5875	10.381	2.3625
3	ZDT3	**0.9187**	3.0265	15.073	1.545
4	ZDT4	**0.6656**	1.8812	5.6312	1.2562
5	ZDT6	**1.215**	2.484	11.435	2.0421
6	SCH1	2.5593	3.0031	4.7359	**1.9218**
7	SCH2	3.843	3.9687	7.6031	**2.8219**
8	BEL	**2.1125**	2.9188	11.9688	2.3688
9	BNH	**4.3250**	5.7266	16.5813	4.6016
10	CF1	**4.9531**	7.7188	21.8906	5.3438
11	CONSTR	4.4063	7.0156	17.1875	**3.4531**
12	OSY	2.375	5.0313	38.875	**1.7813**
13	3-bar truss design	**11.875**	12.5938	47.7188	11.9375
14	Brushless DC motor design	15.7656	28.1094	108.2813	**7.8125**
15	Car crash design	**8.9219**	13.3750	67.9531	10.2031
16	Pressure vessel design	**11.0469**	23.8125	86.4531	14.4844
17	Satellite pipe design	**20.2344**	28.7656	180.4063	15.1563
18	Welded beam design	**6.5**	14.7344	121.8438	6.8281

6. Conclusion

This effort projected a physics-inspired multiobjective approach is imitating the interaction of ions. A storage and leader selection strategy were then combined into a single objective IMO approach to solving multiobjective problems. A group of unconstrained, constrained and engineering benchmark functions were employed to assess the performance of the MOIMO. The outcomes of MOIMO are compared with those of MODA, MOMVO and MOALO. It was evident that the MOIMO algorithm is very competent and modest in determining an exact estimation of Pareto optimal front with uniform dispersal across all objectives with minimum simulation time.

Furthermore, the high convergence of MOIMO results in precisely estimated solutions and the uniform distribution is owing to the excellent exploration. Also, leader selection and storage preservation endorse the spreading of solutions. Examination of various constraint handling methods to solve MOOP using MOIMO can be an excellent future work. Also, the proposed approach can be applied to solving real-world problems.

Acknowledgment
The authors would like to thank Professor Seyedali Mirjalili and Shri Pradeep Jangir for their valuable support.

References

Akbari, R., Hedayatzadeh, R., Ziarati, K., & Hassanizadeh, B. (2012). A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation*, *2*, 39-52.

Buch, H., & Trivedi, I. (2020). A new non-dominated sorting ions motion algorithm: Development and applications. *Decision Science Letters*, *9*(1), 59-76.

Coello, C. A. C. (2011). An introduction to multi-objective particle swarm optimizers. In *Soft computing in industrial applications* (pp. 3-12). Springer, Berlin, Heidelberg.

Coello, C. A. C., & Cortés, N. C. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, *6*(2), 163-190.

Coello, C. C., & Lechuga, M. S. (2002, May). MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1051-1056). IEEE.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, *41*(2), 113-127.

Coello, C. A. C. (2009). Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. *Frontiers of Computer Science in China*, *3*(1), 18-30.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons.

Deb, Kalyanmoy, Samir Agrawal, Amrit Pratap, and T Meyarivan. 2000. "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II." In *Parallel Problem Solving from Nature PPSN VI*, eds. Marc Schoenauer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 849–58.

Dorigo, M., & Di Caro, G. (1999, July). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (Vol. 2, pp. 1470-1477). IEEE.

Javidy, B., Hatamlou, A., & Mirjalili, S. (2015). Ions motion algorithm for solving optimization problems. *Applied Soft Computing*, *32*, 72-79.

Knowles, J., & Corne, D. (1999, July). The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)* (Vol. 1, pp. 98-105). IEEE.

Mirjalili, S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, *27*(4), 1053-1073.

Mirjalili, S., Jangir, P., Mirjalili, S. Z., Saremi, S., & Trivedi, I. N. (2017). Optimization of problems with multiple objectives using the multi-verse optimization algorithm. *Knowledge-Based Systems*, *134*, 50-71.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163-191.

Mirjalili, S., Jangir, P., & Saremi, S. (2017). Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. *Applied Intelligence*, *46*(1), 79-95.

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications*, *27*(2), 495-513.

Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L. D. S. (2016). Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, *47*, 106-119.

Mirjalili, S. Z., Mirjalili, S., Saremi, S., Faris, H., & Aljarah, I. (2018). Grasshopper optimization algorithm for multi-objective optimization problems. *Applied Intelligence*, *48*(4), 805-820.

Ngatchou, P., Zarei, A., & El-Sharkawi, A. (2005, November). Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems* (pp. 84-91). IEEE.

Schott, J. R. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization* (No. AFIT/CI/CIA-95-039). Air Force Inst of Tech Wright-Patterson AFB OH.

Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), 221-248.

Van Veldhuizen, D. A. (1999). Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations. *IRE Transactions on Education*.

Nanda, S. J. (2016, September). Multi-objective moth flame optimization. In *2016 International conference on Advances in computing, communications and informatics (ICACCI)* (pp. 2470-2476). IEEE.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

Zhang, M., Wang, H., Cui, Z., & Chen, J. (2018). Hybrid multi-objective cuckoo search with dynamical local search. *Memetic Computing*, *10*(2), 199-208.

Zou, F., Wang, L., Hei, X., Chen, D., & Wang, B. (2013). Multi-objective optimization using teaching-learning-based optimization algorithm. *Engineering Applications of Artificial Intelligence*, *26*(4), 1291-1300.