Structural and Multidisciplinary Optimization (2023) 66:114
https://doi.org/10.1007/s00158-023-03568-y

RESEARCH PAPER

Multi-objective equilibrium optimizer slime mould algorithm and its application in solving engineering problems

Qifang Luo[1,3] · Shihong Yin[1] · Guo Zhou[2] · Weiping Meng[1,3] · Yixin Zhao[4] · Yongquan Zhou[1,3]

Received: 5 September 2022 / Revised: 16 February 2023 / Accepted: 4 April 2023 / Published online: 24 April 2023

Abstract

This paper aims to represent a multi-objective equilibrium optimizer slime mould algorithm (MOEOSMA) to solve real-world constraint engineering problems. The proposed algorithm has a better optimization performance than the existing multi-objective slime mould algorithm. In the MOEOSMA, dynamic coefficients are used to adjust exploration and exploitation trends. The elite archiving mechanism is used to promote the convergence of the algorithm. The crowding distance method is used to maintain the distribution of the Pareto front. The equilibrium pool strategy is used to simulate the cooperative foraging behavior of the slime mould, which helps to enhance the exploration ability of the algorithm. The performance of MOEOSMA is evaluated on the latest CEC2020 functions, eight real-world multi-objective constraint engineering problems, and four large-scale truss structure optimization problems. The experimental results show that the proposed MOEOSMA not only finds more Pareto optimal solutions, but also maintains a good distribution in the decision space and objective space. Statistical results show that MOEOSMA has a strong competitive advantage in terms of convergence, diversity, uniformity, and extensiveness, and its comprehensive performance is significantly better than other comparable algorithms.

Keywords Multi-objective equilibrium optimizer slime mould algorithm · CEC2020 functions · Real-world engineering problems · Truss structure optimization · Metaheuristic algorithm

1 Introduction

In the past decades, metaheuristic algorithms have been favored by many researchers and successfully applied to solve real-world optimization problems in various fields due to their strong search capability, low computational complexity and strong generalization capability. Researchers have made a lot of efforts to find reasonable and effective search operators for single-objective optimization problems (SOPs), to improve the search capability of metaheuristic algorithms, and to balance the exploration and exploitation of algorithms, and have achieved more satisfactory results. Single-objective optimization algorithms have well solved problems such as cluster analysis (Cui et al. 2022), forest fire rescue (Chen et al. 2022) and crack identification (Benaissa et al. 2021). However, in the real world, many problems are usually composed of multiple conflicting objective functions, and they are called multi-objective optimization problems (MOPs). Since MOPs are widespread in the real world and difficult to be solved efficiently, algorithms for dealing with MOPs have gradually become a popular research topic (Zeng et al. 2021). Taking the minimization problem as an example, the mathematical definition of MOPs is presented in Eq. (1).

Responsible editor: Mehmet Polat Saka.

✉ Yongquan Zhou
yongquanzhou@126.com

[1] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China

[2] Department of Science and Technology Teaching, China University of Political Science and Law, Beijing 100088, China

[3] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China

[4] School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China

$$\begin{aligned} &\text{minimize } F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_M(\mathbf{x})] \\ &\text{subject to } g_i(\mathbf{x}) \le 0,\ i = 1, 2, \cdots, l \\ &h_j(\mathbf{x}) = 0,\ j = 1, 2, \cdots, c \\ &\text{with } lb \le x_k \le ub, k = 1, 2, \cdots, d. \end{aligned} \tag{1}$$

where $\mathbf{x} = [x_1, x_2, \cdots, x_d]^{\mathrm{T}}$ is the vector of decision variables, d is the number of decision variables, $f_i : \mathbb{R}^d \to \mathbb{R}, i = 1, 2, \cdots, M$ are the objective functions, M is the number of objectives, $\mathbb{R}^d$ is the d-dimensional solution space $g_i : \mathbb{R}^d \to \mathbb{R}, i = 1, 2, \cdots, l$ are the inequality constraints functions, l is the number of inequality constraints, $h_i : \mathbb{R}^d \to \mathbb{R}, i = 1, 2, \cdots, c$ are the equality constraints functions, c is the number of equality constraints, lb is the lower bound of decision variables, and ub is the upper bound.

Unlike SOPs, MOPs do not have a global or single optimal solution, but rather have many solutions alternatively representing the optimal solution. In order to compare the superiority of solutions during the iteration process and to provide search agents with environmental selection pressure, the related concept of Pareto dominance is introduced (Coello 2009).

Definition 1 (Pareto dominance) Given two solutions $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, if $\forall i \in \{1, 2, \cdots, M\}$, $f_i(\mathbf{x}) \le f_i(\mathbf{y})$, and $\exists i \in \{1, 2, \cdots, M\}$, such that $f_i(\mathbf{x}) < f_i(\mathbf{y})$, then solution $\mathbf{x}$ is said to dominate $\mathbf{y}$, denoted as $\mathbf{x} \prec \mathbf{y}$.

Definition 2 (Pareto optimality) If $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^d$, $\forall \mathbf{y} \in \mathbf{X}$, $\mathbf{x} \prec \mathbf{y}$, then solution $\mathbf{x}$ is called the Pareto optimality in the solution set $\mathbf{X}$.

Definition 3 (Pareto set) Pareto set (PS) is the set of all Pareto optimal solutions in the solution set $\mathbf{X} \subset \mathbb{R}^d$.

Definition 4 (Pareto front) Pareto front (PF) is the mapping set of the Pareto set (PS) in the objective space.

A challenging task is the need to consider multiple conflicting objective functions simultaneously and then find the best possible trade-off solution that satisfies all constraints, if any. According to definition 1, the non-dominated solution should be strictly better than all solutions for one subfunction and not the best solution for the other subfunctions. The easiest way to deal with MOPs is to assign a specific weight to each subfunction, converting multiple objective functions into a single objective. In practice, this method does not necessarily exist the attainability of design variables for all the objective functions, which means that the objective function may not represent PF correctly and competitiveness may lead to an inappropriate Pareto optimal solution (Ali and Shimoda 2022). An appropriate PF contains a solution $\mathbf{x}$ such that for each subobjective function f_i, there exists at least one f_j such that $f_j(\mathbf{x}) < f_j(\mathbf{y})$, $(f_i(\mathbf{y}) - f_i(\mathbf{x})) \big/ (f_j(\mathbf{x}) - f_j(\mathbf{y})) \le \varepsilon$, where $\mathbf{y} \in \mathbf{X} \subset \mathbb{R}^d$ and ε is a scalar larger than 0.

Generally, there are three crucial factors need to be considered when designing multi-objective optimization algorithms (MOAs): convergence, diversity, and spread (Li and Zheng 2009). Convergence reflects the distance between the obtained PF and the true PF; diversity reflects whether the points on the obtained PF are evenly spaced; and spread reflects the distribution range of the obtained PF. It is very challenging to ensure that MOAs find a evenly spaced and highly convergent PF when solving MOPs (Zhao et al. 2022). MOAs can be classified into three categories based on the participation of decision makers in in the optimization process: the priori method (Jin et al. 2001), posteriori method (Branke et al. 2004), and interactive method (Kollat and Reed 2007). In the priori method, the decision maker provides preference weights in advance, so that multiple objectives can be combined into a single objective through weight allocation. In the posteriori method, the decision maker makes a decision at the end of the optimization and therefore needs to generate a set of alternative non-dominated solutions under conflicting objectives. Finally, in the interactive method, the decision maker needs to participate in the optimization process, so this treatment is less efficient. Compared with the other two methods, the posteriori method maintains the formulation of MOPs, has a stronger randomness, does not require too much decision maker intervention, does not depend on specific problems, and has a stronger generalization capability. Based on these advantages, the posteriori method has become the most popular multi-objective optimization processing method, and many multi-objective biologically inspired algorithms belong to this method. The posteriori method usually employs the concept of Pareto dominance to evaluate the advantages and disadvantages of non-dominated solutions, employs a crowding distance mechanism to improve the distribution of the PF, and employs an archive to preserve the optimal Pareto solutions found so far.

When solving MOPs using the multi-objective biologically inspired algorithms, each candidate solution generates a new solution through the search operator of the algorithm. Then the better non-dominated solutions are then selected to update the archive. There are three main methods for archive updating: indicator-based, decomposition-based, and Pareto-based. Indicator-based MOAs use the performance

indicators (generational distance (GD) (Ayala et al. 2017), inverted generational distance (IGD) (Champasak et al. 2020) and hypervolume (HV) (Gong et al. 2020) etc.) to guide the direction of evolution. Indicator-based evolutionary algorithm (IBEA) (Zitzler and Künzli 2004) is one of the most famous algorithms, whose main idea is to formalize preferences by successive generalizations of the dominance relation, so that two solutions can be directly compared by the designed indicators. The performance of IBEA depends on the indicator designed for a certain class of problems, and its generalization capability is weak. Decomposition-based MOAs decompose the problem into a set of single-objective subproblems and optimize each subproblem using neighborhood information. Decomposition-based multi-objective evolutionary algorithm (MOEA/D) (Zhang and Li 2007) is a representative algorithm among them. MOEA/D utilizes an objective aggregation strategy to decompose MOPs into multiple single-objective subproblems, and each subproblem can be optimized by simply combining the information of the remaining neighboring subproblems, which reduces the computational cost. However, breakpoints can lead to inefficiency of the MOEA/D strategy when solving some MOPs with discontinuous PFs (Qi et al. 2014). Pareto-based MOAs utilize the Pareto dominance principle to evaluate the proximity between the currently obtained PF and the true PF. Representative algorithms are the non-dominated sorting genetic algorithm (NSGA-II) (Deb et al. 2002), NSGA-III (Deb and Jain 2014), Pareto envelope-based selection algorithm (PESA-II) (Corne et al. 2001), strength Pareto evolutionary algorithm (SPEA2) (Zitzler et al. 2001), multi-objective particle swarm optimizer (MOPSO) (Coello et al. 2004), multi-objective seagull optimization algorithm (MOSOA) (Dhiman et al. 2021), multi-objective water cycle algorithm (MOWCA) (Sadollah et al. 2015), multi-objective grasshopper optimization algorithm (MOGOA) (Mirjalili et al. 2018), multi-objective gradient-based optimizer (MOGBO) (Premkumar et al. 2021a), multi-objective artificial bee colony (MOABC) (Hancer et al. 2015), and so on.

Although researchers have proposed some effective MOAs to solve MOPs, the no free lunch (NFL) theorem (Wolpert and Macready 1997) logically proves that no one algorithm is universally superior in handling all MOPs. That is, there is no universal criterion for trade-offs between multiple objectives for different types of problems. In addition, existing MOAs are usually designed based on the search framework of the most basic single-objective optimization algorithms, and their search operators have weak global search capability in the decision space, which largely affects the convergence performance of the algorithms in the objective space. Therefore, the design of efficient MOAs needs to focus on convergence and diversity in both decision space and objective space.

Biologically inspired algorithms simulate the collaborative foraging behavior of organisms in nature, with strong self-organization and adaptive search capability. Many biological systems are composed of individuals with no intelligence, but these individuals can effectively self-organize into systems that achieve a good balance between efficiency and robustness. Slime mould is a promising organism with strong path planning capability due to its unique oscillatory foraging behavior. Tero et al. (2010) simulated the foraging characteristics of the slime mould to design a mathematical model to map Tokyo subway network. Li et al. (2011) designed two local routing protocols for wireless sensor networks using two different mechanisms in the formation process of slime mould tubular networks. Qian et al. (2013) designed an ant colony system based on slime mould to solve the traveling salesman problem. Becker (2015) developed a slime mould algorithm to solve graph optimization problem. Subsequently, Li et al. (2020) summarized the previous design experience and proposed a general global optimization algorithm named slime mould algorithm (SMA). SMA simulates the positive and negative feedback of slime mold using adaptive weights and has a strong local search capability. Due to its clear structure and easy implementation, SMA has received much attention from scholars since its proposal and has been successfully applied to image threshold segmentation (Naik et al. 2022), photovoltaic parameter extraction (Liu et al. 2021), power system optimization (Wei et al. 2021), job shop scheduling (Wei et al. 2022), optimal economic emission scheduling (Hassan et al. 2021), classification and diagnosis of diseases (Wazery et al. 2021), big data forecasting (Chen and Liu 2020), and other optimization problems. However, SMA has rarely been designed to deal with MOPs, especially in terms of enhancing the performance of the basic SMA.

Premkumar et al. (2021b) designed a multi-objective SMA and evaluated the performance of MOSMA on benchmark functions and engineering problems. Subsequently, Hassan et al. (2021) designed an improved multi-objective SMA based on the sine cosine algorithm and applied it to a multi-objective economic emission dispatch problem. Finally, Houssein et al. (2022) also proposed a multi-objective SMA and analyzed the performance of MOSMA in the decision space on CEC2020 functions. There are three main shortcomings of the existing MOSMA: (1) the existing algorithms do not properly deal with the sorting process in SMA, and the non-dominated sorting does not well combined with the fitness weight, leading to poor convergence accuracy of the algorithm on complex problems; (2) the crowding

distance mechanism does not maintain the diversity of solutions in the archive well, leading to poor distribution of the PF. (3) The global search capability of SMA is not improved, which makes the exploration of MOSMA in the decision space inadequate.

Due to the weak exploration capability of SMA, existing MOSMAs usually employ the non-dominated sorting strategy in NSGA-II to construct the archive. This mainly enhances the global search of the algorithm in the objective space, while not improving the algorithm from the decision space. When dealing with complex MOPs, it not only leads to slow convergence of MOSMA, but also easy to fall into local optimum. To improve the search capability of the algorithm in the decision space, Yin et al. (2022b) designed an equilibrium optimizer slime mould algorithm (EOSMA) and revealed the efficiency of the algorithm by comparing it with the CEC winner. The significant advantages of EOSMA on SOPs motivate us to propose its multi-objective version (MOEOSMA) to solve more real-world MOPs. The main contributions of this study are as follows:

1) The elite archive component is applied to EOSMA, which can store the Pareto optimal solutions found so far.
2) The equilibrium pool and crowding distance method are applied to EOSMA to retain the diversity of Pareto optimal solutions.
3) The constant factors in MOEOSMA are replaced by dynamic exploration and exploitation coefficients to enhance the exploration and exploitation.
4) The effectiveness of MOEOSMA was verified in the CEC2020 functions and compared with nine advanced multi-objective algorithms.
5) The convergence results of MOEOSMA are compared with eleven algorithms on eight real-world engineering problems and four truss optimization problems, and the performance is evaluated using the Hypervolume and Spacing-to-Extent indicators.

The rest of this paper is organized as follows. Section 2 briefly introduces EOSMA. Section 3 describes the optimization principle and implementation of MOEOSMA. Section 4 analyzes the performance of MOEOSMA on CEC2020 functions in the decision space and objective space. Section 5 evaluates the efficiency of MOEOSMA on real-world engineering problems and truss optimization problems. Finally, Sect. 6 concludes this paper.

2 Related work

Recently, Yin et al. (2022b) proposed an efficient hybrid equilibrium optimizer slime mould algorithm (EOSMA) and verified the algorithm's performance on CEC2019, CEC2021 test functions, and many real-world engineering optimization problems. In EOSMA, the EO search operator (Faramarzi et al. 2020) is used to guide the oscillatory foraging behavior of slime mould, which makes the anisotropic search of slime mould have a specific orientation, resulting in a better balance between exploration and exploitation. Moreover, the random differential mutation operator and greedy selection strategy are integrated into the algorithm's search process to enhance the exploration and exploitation capabilities simultaneously. The search principle of EOSMA is described as follows.

2.1 Population initialization and equilibrium pool

The first phase in the swarm intelligence optimization algorithm is population initialization. EOSMA uses the randomly generated uniformly distributed positions in the search space as the population's initial solution. The initialization formula is shown in Eq. (2).

$$\mathbf{X}_i = \mathbf{LB} + \mathbf{r} \cdot (\mathbf{UB} - \mathbf{LB}) \tag{2}$$

where $\cdot$ indicates Hadamard product, $\mathbf{LB} = [lb_1, lb_2, \cdots, lb_d]$ is the lower bound of variables, $\mathbf{UB} = [ub_1, ub_2, \cdots, ub_d]$ is the upper bound, $\mathbf{X}_i$ represents the ith initial solution, and $\mathbf{r}$ is the random number vector between [0,1].

To further balance the exploration and exploitation capability, EOSMA integrates an equilibrium pool. The equilibrium pool contains five positions, the first four of which are locally optimal positions to help exploration and the last one of which is their average position to help exploitation. The equilibrium pool is shown in Eq. (3).

$$\mathbf{X}_{eq} = \left[\mathbf{x}_{eq,1}, \mathbf{x}_{eq,2}, \mathbf{x}_{eq,3}, \mathbf{x}_{eq,4}, \mathbf{x}_{eq,avg}\right]^{\mathrm{T}} \tag{3}$$

where $\mathbf{x}_{eq,1}, \mathbf{x}_{eq,2}, \mathbf{x}_{eq,3}, \mathbf{x}_{eq,4}$ denotes the four individuals with the best fitness in the current population, $\mathbf{x}_{eq,avg} = (\mathbf{x}_{eq,1} + \mathbf{x}_{eq,2} + \mathbf{x}_{eq,3} + \mathbf{x}_{eq,4})/4$.

In the iterative process, a position is randomly selected from the equilibrium pool $\mathbf{X}_{eq}$ as the best food source $\mathbf{X}_b$ to guide the updating direction of the search agent, enabling slime mould to utilize multiple food sources at the same time.

2.2 Optimization process of EOSMA

According to the foraging behavior of slime mould, the EOSMA optimization process can be divided into anisotropic search and vein-like tube formation stages. The anisotropic search stage is replaced by the search operator of EO and the venous tube formation stage is updated by the search operator of SMA. The optimization process can be expressed as Eq. (4).

$$\mathbf{X}_i^* = \begin{cases} \mathbf{X}_b + \left(\mathbf{X}_i - \mathbf{X}_b\right) \cdot \mathbf{F} + (\mathbf{G}./\boldsymbol{\lambda}) \cdot (1 - \mathbf{F}) & r_1 < z \\ \mathbf{X}_i + \mathbf{vb} \cdot \left(\mathbf{W}_i \cdot \mathbf{X}_{R1} - \mathbf{X}_{R2}\right) & r_2 < p \\ \mathbf{X}_b + \mathbf{vc} \cdot \left(\mathbf{W}_i \cdot \mathbf{X}_{R1} - \mathbf{X}_{R2}\right) & others \end{cases} \tag{4}$$

where $\cdot$ indicates Hadamard product,./ indicates the division of the corresponding elements of the matrix, $\mathbf{X}_i^*$ denotes the updated ith solution, $i = 1, 2, \cdots, N$, N is the population size, $\mathbf{X}_i$ denotes the current solution, $\mathbf{X}_b$ is a solution randomly selected from the equilibrium pool, $\mathbf{F}$ is the exponential term coefficient, $\mathbf{F} = a_1 sign(\mathbf{r} - 0.5) \cdot \left(e^{-\boldsymbol{\lambda} t_1} - 1\right)$, $t_1 = (1 - t/\max\_t)^{(a_2 t/\max\_t)}$, $a_1 = 2$ and $a_2 = 1$ are adaptive parameters that adjust the exploration and exploitation, t and $\max\_t$ denote the number of current and maximum iterations, $\mathbf{G}$ is the mass generation rate, $\boldsymbol{\lambda}$ is a random number vector between [0,1], $\mathbf{W}_i$ is the fitness weight of the ith slime mould individual, $\mathbf{vb}$ is a random number vector between $[-a, a]$, $a = \text{atanh}(1 - t/\max\_t)$, $\mathbf{vc}$ decreases linearly from 1 to 0, r_1 and r_2 are random numbers between [0,1], $\mathbf{X}_{R1}$ and $\mathbf{X}_{R2}$ represent two randomly selected solutions from the current population, $z = 0.6$ is a hybrid parameter, $p = \tanh\left|\mathbf{Fit}_i - BF\right|$, $\mathbf{Fit}_i$ denotes the fitness of the ith individual, BF is the best fitness. The mass generation rate $\mathbf{G}$ is calculated as shown in Eq. (5).

$$\mathbf{G} = \begin{cases} 0.5 r_1 \cdot \left(\mathbf{X}_b - \boldsymbol{\lambda} \cdot \mathbf{X}_i\right) \cdot \mathbf{F} & r_2 \geq 0.5 \\ 0 & others \end{cases} \tag{5}$$

where r_1 and r_2 are random numbers between [0,1]. The adaptive weight $\mathbf{W}$ is calculated as shown in Eq. (6).

$$\mathbf{W}(\mathbf{FitIdx}_i) = \begin{cases} 1 + \mathbf{r} \cdot \log\left(\frac{bF - \mathbf{Fit}_i}{bF - wF} + 1\right) & i < \frac{N}{2} \\ 1 - \mathbf{r} \cdot \log\left(\frac{bF - \mathbf{Fit}_i}{bF - wF} + 1\right) & others \end{cases} \tag{6}$$

$$\mathbf{FitIdx} = sort(\mathbf{Fit})$$

where $\mathbf{FitIdx}$ represents fitness sorted in ascending order, $\mathbf{r}$ is the random number vector between [0,1], bF and wF are the best and worst fitness values in the current population, $\mathbf{Fit}_i$ represents the fitness of the ith individual, N is the population size.

After the search agent is updated by Eq. (4), the random difference mutation mechanism and greedy selection strategy are employed to enhance the search agent's exploration and exploitation capability, helping the search agent to escape the local optimum. The mathematical formula of the random difference mutation operator is shown in Eq. (7).

$$\mathbf{X}_i^* = \begin{cases} \mathbf{X}_i + CF\left(\mathbf{LB} + r_3(\mathbf{UB} - \mathbf{LB})\right) \cdot \mathbf{L} & r_4 < q \\ \mathbf{X}_i + SF\left(\mathbf{X}_{R1} - \mathbf{X}_{R2}\right) & others \end{cases} \tag{7}$$

where $CF = (1 - t/\max\_t)^{(a_1 t/\max\_t)}$ is an adaptive compression factor, $\mathbf{L}$ is a vector with elements 0 or 1, SF is a random number between [0.2,1], r_3 and r_4 are random numbers between [0,1], and $q = 0.2$ is a tunable parameter. To avoid invalid searches, after updating the location of the search agent, the solution outside the boundary is updated using the dichotomy method, as shown in Eq. (8).

$$\mathbf{X}_{ij}^* = \begin{cases} (\mathbf{X}_{ij} + ub_j)./2 & \mathbf{X}_{ij}^* > ub_j \\ (\mathbf{X}_{ij} + lb_j)./2 & \mathbf{X}_{ij}^* < lb_j \\ \mathbf{X}_{ij}^* & others \end{cases} \tag{8}$$

where $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, d$, N is the population size, d is the dimension of the decision variable, ub_j and lb_j are the upper and lower bounds of the jth decision variable. After boundary checking, the fitness of each search agent is evaluated, and a greedy selection strategy is applied to retain the better individuals, as shown in Eq. (9).

$$\mathbf{X}_i^* = \begin{cases} \mathbf{X}_i^* & F(\mathbf{X}_i^*) < F(\mathbf{X}_i) \\ \mathbf{X}_i & others \end{cases} \tag{9}$$

where $F(\cdot)$ means to evaluate the fitness of an individual, $\mathbf{X}_i^*$ represents the current individual, and $\mathbf{X}_i$ represents the individual of the previous generation.

In EOSMA, greedy selection strategy and equilibrium pool are introduced. The greedy selection strategy maintains an archive of the same capacity as the population size, which stores the best solutions that each search individual has found so far. After each iteration, the archive is updated by Eq. (9). The greedy strategy also provides search agents with a powerful memory, allowing slime mould to recall successful foraging areas in the past. There are five solutions in the equilibrium pool, four of which are local optimal solutions and the other is the central position of the four local optimal solutions. The equilibrium pool allows slime mould to develop multiple food sources at the same time, increasing the probability of obtaining the global optimal solution. The flow chart and pseudo-code of EOSMA can be referred to (Yin et al. 2022b).

3 Proposed MOEOSMA

3.1 Pareto archive

The purpose of multi-objective optimization is to present decision-makers with a large number of Pareto optimal alternatives. These solutions trade-off between multiple objectives, and there is no dominant relationship between them. To design a multi-objective EOSMA, an archive with a defined maximum capacity, similar to MOPSO (Coello and Lechuga 2002), is first integrated into the algorithm. The Pareto dominance operator is employed during optimization to compare the updated slime mould position to the position in the archive. If the updated slime mould individual is not dominated by the individuals in the archive, it will be preserved; otherwise, it will be discarded. When the number of solutions in the archive exceeds the maximum capacity and is not dominant, the crowding distance and roulette method are used to remove the solutions in the dense region of the population to improve the PF distribution.

3.2 Crowding distance

The crowding distance calculation method determines the selection of the archiving solution and consequently has a significant impact on the diversity of the final PF. Three different crowding distances have been proposed in the literature (Mirjalili et al. 2017b; Zeng et al. 2021; Houssein et al. 2022). By combining the characteristics of existing crowding distances, this study proposes a simple and effective approach for measuring crowding distance. As shown in Fig. 1, the crowding distance of the slime mould individual m represents the number of neighborhood solutions in the hypercube centered on itself. The length of the i^{th} side of the hypercube is $2d_i$, and the distance d_i is defined as Eq. (10).

$$d_i = \frac{f_i^{\max} - f_i^{\min}}{As}, i = 1, 2, \cdots, M \tag{10}$$

where $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of the ith objective function, M is the number of objective functions and As is the archive size.

As shown in Fig. 1, the non-dominated solutions with smaller crowding distances are in sparser regions and are important for approximating the true PF, while the non-dominated solutions with larger crowding distances are in denser regions and have the least influence on the distribution of PS, and are given a higher probability to be removed. Therefore, when the archive is filled, the probability that each solution is removed from the archive is shown in Eq. (11).

$$P_i = \frac{N_i}{C} \tag{11}$$

where C denotes the sum of the crowding distances of all solutions in the archive, and N_i denotes the crowding distance of the ith solution.

In this way, MOEOSMA can store better non-dominated solutions in the archive and constantly improve them during iteratives. Using less crowded solutions as food sources can promote slime mould to find other food sources nearby. This will naturally attract search agents to regions with fewer non-dominated solutions in the PF, improving the coverage of the final obtained PF. It is worth noting that, in contrast to the existing MOSMA (Premkumar et al. 2021b; Houssein et al. 2022), the crowding distance used in this research is a discrete value rather than a continuous value, which not only preserves the randomness of the solution being selected in the archive but also facilitates the selection of elite individuals to simulate the multiple food sources found by slime mould. In MOEOSMA, the solution in the archive with the minimum crowding distance is placed in the equilibrium pool. During the iteration, each slime mould individual randomly selected a solution from the equilibrium pool as the food source $\mathbf{X}_b$ that the current search agent decided to exploit.

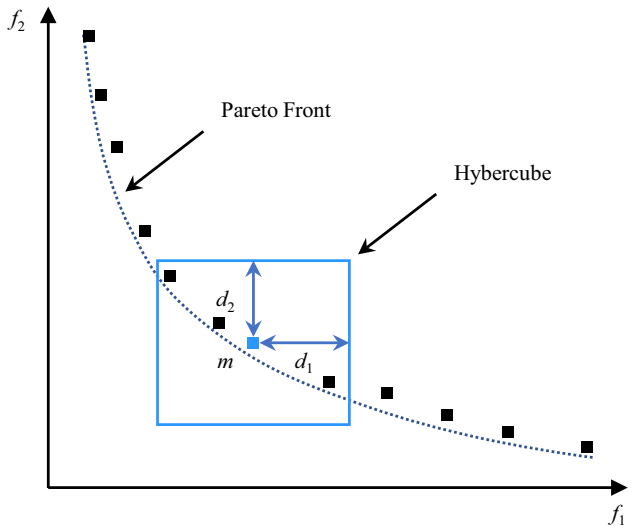


Fig. 1 The crowding distance of the Pareto solution

3.3 Dynamic coefficient

The parameters a_1 and a_2 that adjust the exploration and exploitation in EOSMA are set to fixed values during iteration, which indicates that the algorithm's exploration and exploitation trend will remain unchanged during optimization. Based on the improving experience of many algorithms, it can be concluded that the meta-heuristic algorithm should focus on exploration in the early stage, conducting a broad exploration of the whole search space, and then gradually shift to exploitation, searching for more promising regions. Therefore, during MOEOSMA optimization, a_1 and a_2 are controlled according to the current number of iterations and randomly generated values, which are then translated into dynamic coefficients to adjust the exploration and exploitation trends better. The parameters a_1 and a_2 are calculated as Eq. (12).

$$\begin{aligned} a_1 &= \left(1 + (1 - t/\max\_t)^{2t/\max\_t}\right) \times r \\ a_2 &= \left(2 - (1 - t/\max\_t)^{2t/\max\_t}\right) \times r \end{aligned} \tag{12}$$

where r represents the random number between [0,1], t represents the current iteration number, and max $\_t$ represents the maximum iteration number.

3.4 Optimization process of MOEOSMA

Based on the foraging behavior of slime mould, the EOSMA optimization process may be divided into two stages: the anisotropic search stage and the vein-like tube formation stage. The anisotropic search stage is replaced by the EO search operator with the goal to guide the search direction of the search agents, expanding the search range, and avoiding premature convergence. The vein-like tube formation stage is updated by SMA's most crucial search operator. It is worth noting that slime mould in two stages exists in the population simultaneously, as shown in Eq. (13).

$$\mathbf{X}_i^* = \begin{cases} \mathbf{X}_b + \left(\mathbf{X}_i - \mathbf{X}_b\right) \cdot \mathbf{F} + (\mathbf{G}./\boldsymbol{\lambda}) \cdot (1 - \mathbf{F}) & r < z \\ \mathbf{X}_b + \mathbf{vb} \cdot \left(\mathbf{W}_i \cdot \mathbf{X}_{R1} - \mathbf{X}_{R2}\right) & others \end{cases} \tag{13}$$

where $\cdot$ indicates Hadamard product,./ indicates the division of the corresponding elements of the matrix, $\mathbf{X}_i^*$ denotes the updated ith solution, $\mathbf{X}_i$ denotes the current solution, $\mathbf{X}_b$ is a solution randomly selected from the equilibrium pool, $\mathbf{F}$ is the exponential term coefficient, $\mathbf{F} = a_1 sign(\mathbf{r} - 0.5) \cdot \left(e^{-\boldsymbol{\lambda} t_1} - 1\right)$, $t_1 = (1 - t/\max\_t)^{(a_2 t/\max\_t)}$, a_1 and a_2 are calculated by Eq. (12), $\mathbf{G}$ is calculated by Eq. (5), $\boldsymbol{\lambda}$ is a random number vector between [0,1], $\mathbf{W}_i$ is the fitness weight of the i^{th} slime mould individual, **vb** is a random number vector between $[-a, a]$, $a = \text{atanh}(1 - t/\max\_t)$, r is a random number between [0,1], $\mathbf{X}_{R1}$ and $\mathbf{X}_{R2}$ represent two randomly selected solutions from the current population, and $z = 0.6$ is an adjustable parameter controlling the balance of exploration and exploitation (Yin et al. 2022b). The adaptive weight $\mathbf{W}$ is calculated as shown in Eq. (14).

$$\mathbf{W}(\mathbf{FitIdx}(i)) = \begin{cases} 1 + \mathbf{r} \cdot \log\left(\frac{bF - \mathbf{Fit}_{ik}}{bF - wF} + 1\right) & i < \frac{N}{2} \\ 1 - \mathbf{r} \cdot \log\left(\frac{bF - \mathbf{Fit}_{ik}}{bF - wF} + 1\right) & others \end{cases} \tag{14}$$

$$\mathbf{FitIdx} = sort(\mathbf{Fit}_k), k = rem(t, M) + 1$$

where $\cdot$ indicates Hadamard product, **FitIdx** represents fitness sorted in ascending order, rem$(\cdot)$ is the remainder function, t is the current iteration number, M is the number of objective functions, $\mathbf{r}$ is the random number vector between [0,1], bF and wF are the best and worst fitness values in the current population, $\mathbf{Fit}_{ik}$ represents the fitness of the k^{th} objective function of the ith individual, and N is the population size.

After updating the location of the slime mould by Eq. (13), the random difference mutation operator and greedy selection strategy are introduced to improve the exploration and exploitation capability, and help the search agent to escape the local optimum and obtain the solution with higher accuracy. The mathematical formula of the mutation operator is shown in Eq. (15).

$$\mathbf{X}_i^* = \mathbf{X}_i + SF\left(\mathbf{X}_{R1} - \mathbf{X}_{R2}\right) \tag{15}$$

where $\mathbf{X}_i^*$ denotes the updated ith solution, $\mathbf{X}_i$ denotes the current solution, SF is a random number between [0.2,1], $\mathbf{X}_{R1}$ and $\mathbf{X}_{R2}$ represent two randomly selected solutions from the current population.

To avoid invalid searches, after the location of slime mould is updated, check whether $\mathbf{X}^*$ is beyond the search range and update the location using Eq. (8). Then the fitness is evaluated, and the greedy selection strategy, as shown in Eq. (9), is implemented to retain the better slime mould individuals. The pseudo-code of MOEOSMA is presented in Algorithm 1, and the flow chart is displayed in Fig. 2.

Algorithm 1 Pseudo-code of MOEOSMA

Input the parameters $z, N, d, \max\_t, As$

Output the PS and its fitness PF

Initialize the locations of the search agent $\mathbf{X}_i (i = 1, 2, \cdots, N)$;
while $t \le max\_t$
    Check the boundary by Eq. (8) and calculate the fitness;
    Save the Pareto optimal solutions to the archive;
    Rank the PS based on the crowding distance;
    if *Archive overflow*
        Remove crowded solutions by the roulette method;
        Rank the PS based on the crowding distance;
    end if
    Put the solution with the smallest crowding distance into the equilibrium pool;
    Retain better solutions by Eq. (9);
    Calculate the fitness weight of the slime mould by Eq. (14);
    Calculate the values of adaptive variables a_1, a_2, t_1, a;
    Update the random variables $\lambda, r, r_1, r_2, r_n, vb$;
    for $i = 1$ to N
        Select a solution as the best food source $\mathbf{X}_b$ from the equilibrium pool;
        if $rand < z$
            Update $\mathbf{X}_i^*$ using the EO operator;
        else
            Update $\mathbf{X}_i^*$ using the SMA operator;
        end if
    end for
    Check the boundary by Eq. (8) and calculate the fitness;
    Save the Pareto optimal solutions to the archive;
    Rank the PS based on the crowding distance;
    if *Archive overflow*
        Remove more crowded solutions by roulette method;
    end if
    Retain better solutions by Eq. (9);
    Update $\mathbf{X}_i^*$ by Eq. (15);
    $t = t + 1$;
end while

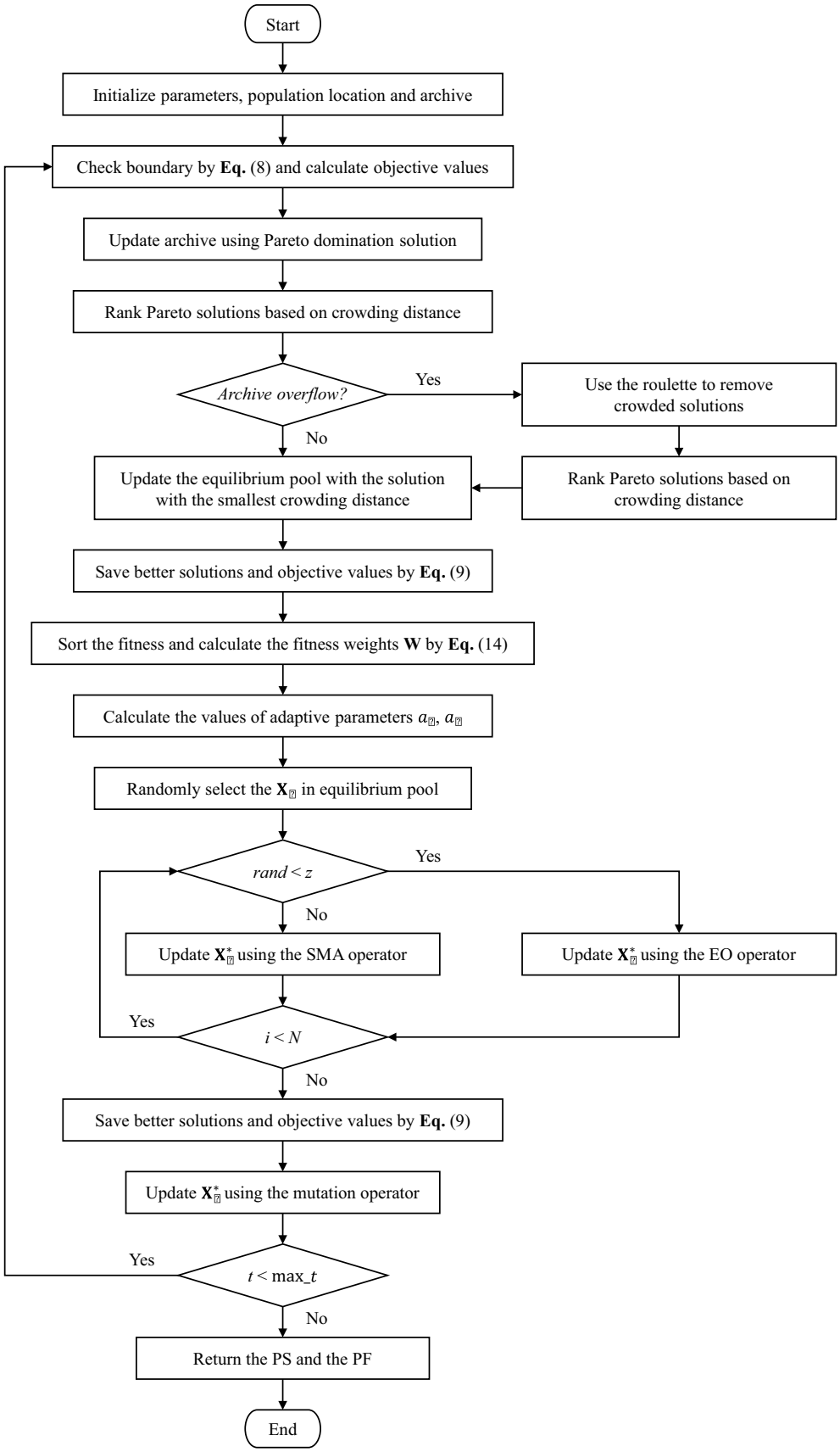


Fig. 2 Flow chart of the MOEOSMA

Table 1 Parameter settings of the comparison algorithms

Algorithms	Parameters	Values	Algorithms	Parameters	Values
MOEOSMA	Hybrid parameter z	0.6	MOEA/D	Crossover parameter γ	0.5
	Generation probability GP	0.5	MOPSO	Inertia weight w	0.5
MOSMA	Constant z	0.03		Damping rate wd	0.99
MOGWO	Grid inflation rate α	0.1		Personal cognition coefficient c_1	1
	Number of grids n	10		Social cognition coefficient c_2	2
	Leader selection pressure β	4		Number of grids n	7
	Deletion selection pressure γ	2		Grid inflation rate α	0.1
MOMVO	Minimum probability of wormhole existence	0.2		Leader selection pressure β	2
	Maximum probability of wormhole existence	1		Deletion selection pressure γ	2
PESA-II	Number of grids n	7		Mutation rate μ	0.1
	Inflation factor	0.1	SPEA2	Crossover parameter p	0.7
	Leader selection pressure β	2		Crossover parameter γ	0.1
	Deletion selection pressure γ	1		Mutation parameter h	0.2
	Crossover parameter p	0.5	MOALO	Parameter less	NA
	Crossover parameter γ	0.15	MSSA	Parameter less	NA
	Mutation parameter h	0.3			

3.5 Computational complexity

The proposed MOEOSMA is mainly made up of the following subcomponents: initialization, position update, fitness evaluation, fitness sorting, equilibrium pool update, fitness weight update, greedy selection, archive update, and mutation operator. Initialization, position update, mutation operation, and fitness weight update all have $O(N * d)$ time complexity, fitness sorting has $O(N * \log N)$ time complexity, greedy selection and equilibrium pool update have $O(N)$ time complexity, and archive update has $O(As^2 * M)$ computational complexity. Therefore, the total time complexity of the algorithm is $O(\max\_t * (N * d + N * \log N + F + As^2 * M))$, where F is the evaluation time of the fitness function, N is the population size, As is the archive size, M is the number of objectives, d is the dimensionality of the problem, and $\max\_t$ is the maximum number of iterations. The space complexity is $O(N * d)$.

4 Experimental and analysis of test functions

In order to verify the effectiveness of the proposed MOEOSMA, the CEC2020 functions are used to analyze the convergence behavior of the algorithm in the objective space and decision space. Unlike previous test suites, CEC2020 includes not only the true PF for each test problem, but also the associated local and global PSs, allowing researchers to evaluate the algorithm's performance in both the objective space and decision space. The MATLAB code of CEC2020 benchmark function can be downloaded at https://github.com/P-N-Suganthan. The specifics of these test functions were described in (Yue et al. 2019; Liang et al. 2020).

4.1 Experimental setup

To evaluate the performance of MOEOSMA relative to other competing algorithms, it is compared to nine well-known MOAs: multi-objective slime mould algorithm (MOSMA) (Premkumar et al. 2021b), multi-objective ant lion optimizer (MOALO) (Mirjalili et al. 2017c), multi-objective grey wolf optimizer (MOGWO) (Mirjalili et al. 2016), multi-objective multi-verse optimization (MOMVO) (Mirjalili et al. 2017b), multi-objective particle swarm optimizer (MOPSO) (Coello et al. 2004), multi-objective Salp swarm algorithm (MSSA) (Mirjalili et al. 2017a), MOEA/D (Zhang and Li 2007), PESA-II (Corne et al. 2001), SPEA2 (Zitzler et al. 2001). The source codes of the comparison algorithms used in the experiments are available on websites: https://aliasgharheidari.com/SMA.html, https://seyedalimirjalili.com, and https://yarpiz.com. All algorithms are executed in MATLAB R2020b under Win 10 OS with hardware details: AMD A8-7410 APU, AMD Radeon R5 Graphics (2.20 GHz) and 12 GB RAM. For a fair comparison, the population size and archiving capacity of all comparison algorithms are set to $200 \times N\_ops$, with a maximum of $10000 \times N\_ops$ evaluations, and 21 independent runs, where N_ops is the number of local and global PS. The other algorithm parameters set in the original paper are listed in Table 1.

Table 2 The IGDX values obtained by all comparison algorithms

Functions	Index	MOEOSMA	MOSMA	MOALO	MOGWO	MOMVO	MOPSO	MSSA	MOEA/D	PESA-II	SPEA2
MMF1	Mean	**0.04568**	0.09669	0.17541	0.22136	0.17675	0.11012	0.24407	0.22476	0.15678	0.10916
	Std.	**0.00479**	0.01707	0.02889	0.06048	0.03790	0.01706	0.04551	0.04807	0.02437	0.02542
MMF2	Mean	**0.02009**	0.12164	0.12203	0.12385	0.13887	0.16046	0.12860	0.27163	0.12670	0.07141
	Std.	**0.01820**	0.05735	0.05459	0.08927	0.09981	0.07051	0.06127	0.10658	0.07252	0.04237
MMF4	Mean	**0.03325**	0.03919	0.17765	0.39771	0.37970	0.10995	0.32256	0.24595	0.12352	0.15427
	Std.	**0.00395**	0.00911	0.06583	0.08519	0.05643	0.03373	0.04582	0.06241	0.03840	0.06285
MMF5	Mean	**0.08098**	0.14957	0.32125	0.57291	0.33741	0.17858	0.49814	0.36972	0.22466	0.19295
	Std.	**0.01007**	0.01836	0.06163	0.11151	0.09744	0.02889	0.07219	0.06586	0.04293	0.03555
MMF7	Mean	**0.03198**	0.05566	0.17005	0.25259	0.19958	0.06824	0.19903	0.13793	0.10601	0.08528
	Std.	**0.00892**	0.00906	0.04268	0.07081	0.04439	0.01251	0.04184	0.03885	0.02296	0.03398
MMF8	Mean	**0.12931**	0.19158	0.84525	1.84536	1.24908	0.29753	1.26859	1.05881	1.97001	2.81676
	Std.	0.09608	**0.08868**	0.59322	1.03616	0.34925	0.18238	0.27480	0.32374	0.64195	0.54080
MMF10	Mean	0.01956	0.05831	0.11536	0.02871	0.31266	0.17673	0.20098	0.33210	0.01552	**0.01403**
	Std.	0.07846	0.02726	0.07306	0.04704	0.09406	0.17344	0.09860	0.10711	0.02487	**0.01359**
MMF11	Mean	**0.00424**	0.01706	0.02153	0.01209	0.01502	0.00861	0.01555	0.01486	0.00975	0.00630
	Std.	**0.00032**	0.00504	0.00813	0.00336	0.00246	0.00093	0.00212	0.00629	0.00341	0.00117
MMF12	Mean	**0.00203**	0.01169	0.01179	0.00421	0.00599	0.00536	0.01054	0.01465	0.00294	0.00245
	Std.	**0.00022**	0.00257	0.00720	0.00159	0.00200	0.00148	0.00286	0.01597	0.00067	0.00071
MMF13	Mean	**0.04459**	0.07220	0.10636	0.15352	0.08925	0.10730	0.10061	0.16144	0.14352	0.12418
	Std.	0.01252	**0.00796**	0.03104	0.03215	0.01746	0.05151	0.01856	0.04130	0.04604	0.03526
MMF14	Mean	0.06605	**0.06376**	0.13791	0.35524	0.29699	0.18942	0.18300	0.17267	0.16975	0.22438
	Std.	0.03387	**0.00858**	0.02520	0.03919	0.01610	0.07775	0.02895	0.04110	0.02526	0.03629
MMF15	Mean	**0.04690**	0.07321	0.09927	0.17561	0.10111	0.06325	0.10948	0.07530	0.07146	0.06930
	Std	**0.00223**	0.01248	0.00770	0.02494	0.02554	0.00267	0.01108	0.00720	0.00764	0.00439
MMF1_e	Mean	2.83622	**1.34712**	1.53306	2.62735	3.02363	1.81669	2.41909	3.33013	2.38024	3.08057
	Std.	0.52038	**0.52004**	0.68605	0.94419	0.53170	0.87476	0.66831	0.62876	0.98143	0.56592
MMF14_a	Mean	**0.08017**	0.12783	0.17652	0.29992	0.30352	0.15027	0.26688	0.19061	0.23093	0.14412
	Std.	**0.00830**	0.02262	0.02443	0.02227	0.08038	0.00961	0.03372	0.02069	0.05444	0.02987
MMF15_a	Mean	0.09106	**0.08181**	0.12028	0.14691	0.14468	0.08935	0.15058	0.10687	0.10409	0.08637
	Std.	0.02127	0.01082	0.01486	0.02460	0.05469	0.00889	0.01289	0.01784	0.01521	**0.00597**
MMF10_l	Mean	0.19891	0.19530	0.18297	0.21067	0.16555	0.20190	**0.14079**	0.17413	0.20504	0.20271
	Std.	0.00757	0.04187	0.06674	0.01440	0.04846	0.00435	0.04226	0.03619	0.01157	**0.00050**
MMF11_l	Mean	0.24957	0.24850	0.25893	0.25533	0.25587	0.25150	0.25271	**0.24538**	0.25770	0.25268
	Std.	**0.00025**	0.00130	0.00268	0.00133	0.00207	0.00107	0.01608	0.03313	0.00458	0.00074
MMF12_l	Mean	0.24515	0.24204	0.24995	0.24628	0.24734	0.24477	0.25316	0.24982	0.24780	**0.24070**
	Std.	0.00166	0.00842	0.00182	0.00123	**0.00073**	0.00131	0.01220	0.01469	0.00176	0.02068
MMF13_l	Mean	**0.26837**	0.26901	0.32531	0.37788	0.31235	0.31054	0.31638	0.34235	0.34269	0.34640
	Std.	0.00635	**0.00485**	0.02715	0.01900	0.02493	0.04928	0.01862	0.02152	0.04216	0.02624
MMF15_l	Mean	0.26110	**0.24068**	0.27371	0.36647	0.30050	0.27829	0.31077	0.29119	0.26662	0.27070
	Std.	**0.00183**	0.02218	0.02447	0.01778	0.00947	0.00412	0.01976	0.00383	0.02083	0.01257
MMF15_a_l	Mean	**0.20894**	0.22855	0.24588	0.31408	0.28524	0.24878	0.29958	0.25261	0.24976	0.24030
	Std.	0.01221	**0.00695**	0.01608	0.03070	0.04837	0.01287	0.02101	0.00831	0.01087	0.00931
MMF16_l1	Mean	0.15864	**0.15740**	0.20755	0.39586	0.28252	0.24240	0.24414	0.22639	0.23062	0.21505
	Std.	0.01246	0.00923	0.01358	0.02929	0.04364	0.05353	0.01529	**0.00822**	0.03045	0.01976
MMF16_l2	Mean	0.33235	**0.30008**	0.32159	0.43202	0.37020	0.34790	0.35706	0.36399	0.34593	0.33437
	Std.	**0.00110**	0.03031	0.04109	0.01490	0.00921	0.00365	0.02926	0.00362	0.01950	0.01697
MMF16_l3	Mean	0.21376	**0.20115**	0.24699	0.46645	0.35960	0.27702	0.28683	0.27133	0.26874	0.25626
	Std.	0.01557	**0.00880**	0.01217	0.01438	0.06001	0.04621	0.01230	0.01037	0.02452	0.02044
FAR (Rank)		**1.77 (1)**	2.92 (2)	6.10 (6)	8.46 (10)	7.60 (9)	4.48 (4)	7.40 (8)	6.56 (7)	5.83 (5)	3.88 (3)

*Bold indicates the optimal result, and FAR stands for Friedman's average ranking

Table 3 The IGDF values obtained by all comparison algorithms

Functions	Index	MOEOSMA	MOSMA	MOALO	MOGWO	MOMVO	MOPSO	MSSA	MOEA/D	PESA-II	SPEA2
MMF1	Mean	**0.00193**	0.00750	0.01637	0.01779	0.01307	0.00909	0.02428	0.01958	0.01450	0.00666
	Std.	**0.00011**	0.00180	0.00353	0.00403	0.00203	0.00135	0.00246	0.00723	0.00361	0.00054
MMF2	Mean	**0.00233**	0.02382	0.03266	0.01364	0.01460	0.07346	0.03091	0.09244	0.02710	0.01243
	Std.	**0.00017**	0.00919	0.00955	0.00485	0.00519	0.02781	0.00792	0.04337	0.00422	0.00515
MMF4	Mean	**0.00171**	0.00194	0.01673	0.01240	0.01721	0.00873	0.02787	0.00919	0.01115	0.00689
	Std.	0.00013	**0.00013**	0.00395	0.00288	0.00217	0.00198	0.00355	0.00160	0.00221	0.00051
MMF5	Mean	**0.00188**	0.00704	0.01458	0.01418	0.01176	0.00892	0.02393	0.01661	0.01249	0.00660
	Std.	**0.00010**	0.00150	0.00251	0.00339	0.00167	0.00145	0.00382	0.00794	0.00443	0.00037
MMF7	Mean	**0.00193**	0.00536	0.02075	0.02169	0.01776	0.00846	0.02462	0.01201	0.01167	0.00686
	Std.	**0.00010**	0.00150	0.00394	0.00291	0.00377	0.00095	0.00516	0.00261	0.00261	0.00058
MMF8	Mean	**0.00171**	0.00520	0.01666	0.00904	0.01339	0.01223	0.02173	0.01193	0.01234	0.00695
	Std.	**0.00013**	0.00281	0.00535	0.00140	0.00163	0.00405	0.00233	0.00637	0.00465	0.00073
MMF10	Mean	**0.02428**	0.18941	0.22412	0.05744	0.32580	0.17114	0.29168	0.30814	0.03968	0.03362
	Std.	0.06343	0.05978	0.07811	0.06138	0.05499	0.14033	0.13370	0.06560	0.03903	**0.01669**
MMF11	Mean	**0.01415**	0.05292	0.10159	0.04883	0.05752	0.03394	0.07297	0.23581	0.04027	0.02900
	Std.	**0.00122**	0.01418	0.01904	0.01084	0.00656	0.00281	0.01420	0.12167	0.00781	0.00301
MMF12	Mean	**0.00309**	0.02965	0.02856	0.00919	0.01238	0.01062	0.02341	0.03306	0.00794	0.00585
	Std.	**0.00024**	0.01070	0.01075	0.00328	0.00399	0.00266	0.00484	0.02211	0.00105	0.00069
MMF13	Mean	**0.01741**	0.04184	0.12781	0.06521	0.07972	0.04126	0.08631	0.31974	0.04698	0.03644
	Std.	**0.00099**	0.02342	0.03164	0.01241	0.01296	0.00405	0.01921	0.15337	0.00742	0.00413
MMF14	Mean	**0.07094**	0.09027	0.17354	0.46218	0.21935	0.12537	0.22795	0.15629	0.14666	0.14292
	Std.	**0.00370**	0.01556	0.01370	0.09706	0.03845	0.00456	0.03465	0.00794	0.01165	0.01352
MMF15	Mean	**0.09520**	0.16077	0.21787	0.45008	0.22548	0.12583	0.23444	0.15988	0.14943	0.14481
	Std.	**0.00331**	0.02901	0.02382	0.09595	0.05824	0.00367	0.02915	0.00954	0.01416	0.01170
MMF1_e	Mean	**0.00191**	0.04295	0.02558	0.01330	0.01033	0.05322	0.02571	0.04437	0.01720	0.00931
	Std.	**0.00014**	0.01062	0.00862	0.00292	0.00148	0.01492	0.00685	0.02395	0.00603	0.00513
MMF14_a	Mean	**0.08666**	0.10912	0.18215	0.37598	0.21867	0.13464	0.33758	0.15237	0.15712	0.14980
	Std.	0.00580	0.01039	0.02270	0.08946	0.03987	**0.00516**	0.08293	0.00660	0.00923	0.00987
MMF15_a	Mean	0.14403	**0.13333**	0.20944	0.25557	0.21878	0.13945	0.27295	0.17074	0.16228	0.15413
	Std.	0.02073	0.01444	0.01700	0.07785	0.02887	**0.00516**	0.02690	0.01614	0.01200	0.02174
MMF10_l	Mean	**0.18238**	0.25596	0.27901	0.21962	0.27148	0.20226	0.26622	0.28956	0.22620	0.21405
	Std.	0.01419	0.03384	0.04388	0.02127	0.07741	0.01001	0.03502	0.07753	0.02685	**0.00847**
MMF11_l	Mean	**0.09270**	0.10769	0.17669	0.12972	0.14068	0.11432	0.15414	0.36828	0.12814	0.11388
	Std.	**0.00047**	0.00707	0.02330	0.01064	0.01196	0.00349	0.01510	0.17527	0.01300	0.00377
MMF12_l	Mean	**0.07989**	0.09702	0.10207	0.09236	0.08925	0.09160	0.10553	0.11217	0.08970	0.08410
	Std.	0.00842	0.01197	0.00804	0.00863	**0.00194**	0.00539	0.01634	0.01800	0.00906	0.01008
MMF13_l	Mean	**0.14344**	0.15169	0.25928	0.19373	0.20725	0.16619	0.20909	0.58981	0.19072	0.18196
	Std.	0.00636	**0.00394**	0.04829	0.01131	0.01644	0.00649	0.01628	0.23681	0.02808	0.01019
MMF15_l	Mean	**0.18716**	0.20209	0.28770	0.54525	0.30563	0.22519	0.34739	0.25716	0.24020	0.23155
	Std.	**0.00224**	0.01082	0.02625	0.07533	0.03404	0.00587	0.08289	0.00793	0.01152	0.00892
MMF15_a_l	Mean	**0.19521**	0.20127	0.25586	0.34355	0.30386	0.22232	0.36512	0.25769	0.23737	0.22555
	Std.	0.01010	0.00690	0.01852	0.07263	0.04040	**0.00487**	0.07080	0.00987	0.01031	0.00789
MMF16_l1	Mean	**0.14278**	0.15842	0.23937	0.55515	0.28537	0.19451	0.29920	0.23036	0.22352	0.20732
	Std.	**0.00292**	0.01542	0.02336	0.10338	0.03743	0.00319	0.03381	0.01006	0.02238	0.01105
MMF16_l2	Mean	0.23574	**0.23534**	0.30704	0.59855	0.34781	0.27260	0.37058	0.31217	0.29817	0.27412
	Std.	**0.00169**	0.00748	0.02340	0.08433	0.02989	0.00530	0.04268	0.00919	0.02126	0.00989
MMF16_l3	Mean	**0.18270**	0.19416	0.26719	0.58961	0.31717	0.22964	0.34974	0.27311	0.26237	0.24465
	Std.	**0.00456**	0.00893	0.01027	0.06495	0.03861	0.00567	0.05248	0.01038	0.01407	0.01046
FAR (Rank)		**1.46 (1)**	4.02 (4)	7.67 (8)	7.02 (7)	6.19 (6)	3.69 (3)	8.94 (10)	8.02 (9)	5.21 (5)	2.79 (2)

*Bold indicates the optimal result, and FAR stands for Friedman's average ranking

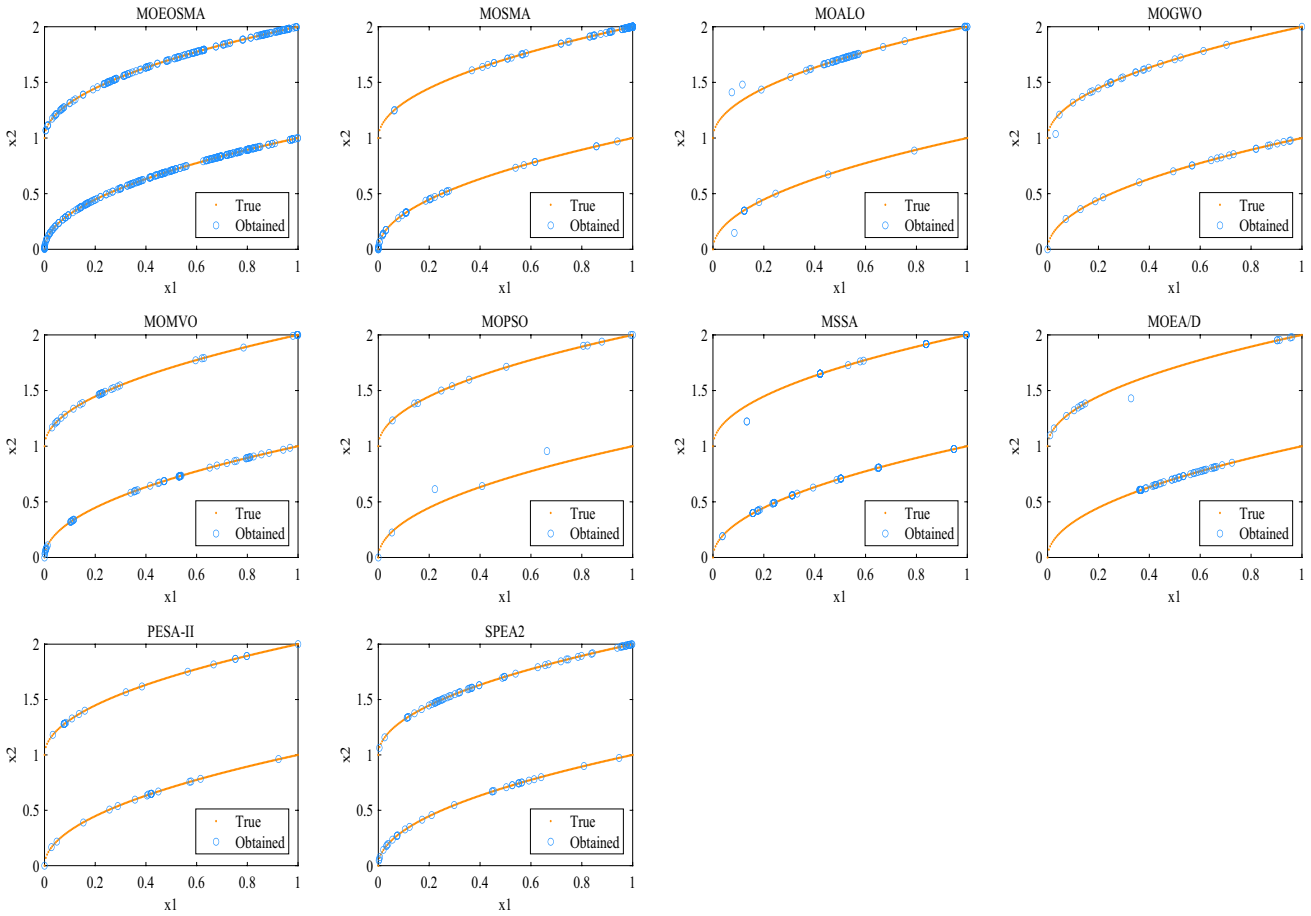


Fig. 3 The optimal PS obtained by all comparison algorithms on MMF2

4.2 Experimental results and analysis

Since CEC2020 functions contain multiple global optimal PSs, a good performance of the algorithm in the objective space does not mean that multiple global optimal PSs can be found. The IGD (Zhang et al. 2008) in decision space (IGDX) (Zhou et al. 2009) and objective space (IGDF) (Zhou et al. 2009) are used to evaluate the quality of the obtained PS and PF, respectively. In the decision space, the smaller the IGDX value, the closer the obtained PS is to the true PS. In the objective space, the smaller the IGDF value, the closer the obtained PF is to the true PF.

The mean and standard deviation of the IGDX obtained by MOEOSMA and comparison algorithms are shown in Table 2. The IGDX value quantifies the convergence of the obtained PS in the decision space. It can be seen from Table 2 that MOEOSMA and MOSMA obtained the minimum values on 12 and 8 functions, respectively, while MSSA, MOEA/D and SPEA2 obtained the best results on a few functions. Friedman's statistical test results reveal that MOEOSMA ranks first, and far better than the ranking values of other comparison algorithms. By comparing the IGDX values, it can be seen that MOEOSMA outperforms other algorithms in search in decision space and is able to find multiple global optimal PSs. It is found that MOEOSMA's superior performance in decision space is mainly due to the equilibrium pool in the EOSMA framework, which stores non-dominated solutions with minimum crowding distance. During the iteration, each slime mould individual randomly selects a solution from the equilibrium pool as the current best food source. This expands the search range of the slime mould in the decision space and enables the algorithm to explore multiple local optimal regions simultaneously. The equilibrium pool strategy not only increases the probability of finding multiple global PSs, but also helps to improve the distribution of PF. In addition, the dynamic exploration and exploitation coefficient improves the search capability of EOSMA, thus improving the convergence accuracy of the algorithm.

Table 3 displays the mean and standard deviation of the IGDF obtained by MOEOSMA and other comparison algorithms. The IGDF reflects the convergence and diversity

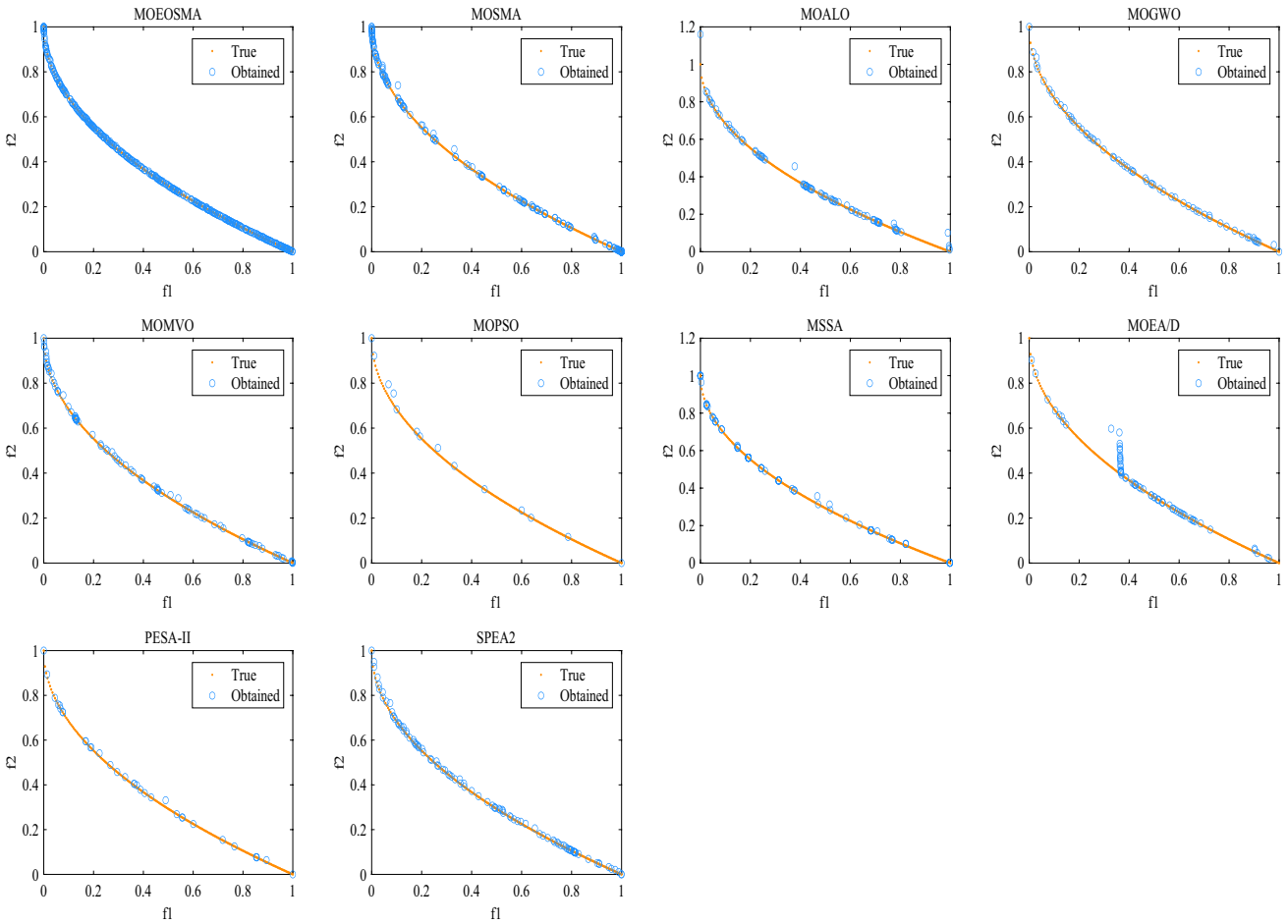


Fig. 4 The optimal PF obtained by all comparison algorithms on MMF2

of the generated PF in the objective space. The results in Table 3 show that MOEOSMA ranks first on 22 functions and does not reach the minimum on MMF15_a and MMF16_l2, but still achieves good results. Combined with the statistical results in Table 2, it can be seen that the performance of MOEOSMA is close to MOSMA in the decision space, but its convergence in the objective space is obviously better than MOSMA. This indicates that the Pareto archive and crowding distance evaluation mechanism used by MOEOSMA are better than MOSMA, which not only improves the convergence rate of PF, but also maintains the diversity of PF well. MOEOSMA differs from MOSMA in terms of archive. Due to the weak exploration capability of SMA, the existing MOSMA (Premkumar et al. 2021b; Houssein et al. 2022) select archived solutions based on the non-dominated level, while MOEOSMA selects archived solutions based on crowding distance by sorting solutions with the highest non-dominated level. The archive of MOSMA is beneficial to exploration, but good non-dominated solutions are easily discarded, while the archive of MOEOSMA can reduce damage to existing archived solutions. As a result, MOEOSMA can provide better convergence than MOSMA and most archive-based MOAs.

Figures 3, 4, 5, and 6 exhibit the best convergence results of PS and PF obtained by all comparison algorithms on MMF2 and MMF16_l3, respectively. These figures display only the non-dominated solutions obtained by the comparison algorithms, not all the solutions of the final population. As shown in Figs. 3 and 4, MOEOSMA obtains more non-dominated solutions on MMF2 and is significantly superior to other algorithms in terms of convergence and distribution. Moreover, Fig. 3 shows that MOEOSMA can also find multiple global PSs, indicating that the algorithm is also suited for handling multimodal multi-objective optimization problems.

MMF16_l3 is a complex three-objective test function with the coexistence of local and global PS. Figures 5 and 6 exhibit the comparison algorithm's search performance on this function. It can be intuitively seen from the figures that MOEOSMA obtains the best results among all the comparison algorithms. It can not only jump out of the local PS but also find multiple uniformly distributed global PSs.

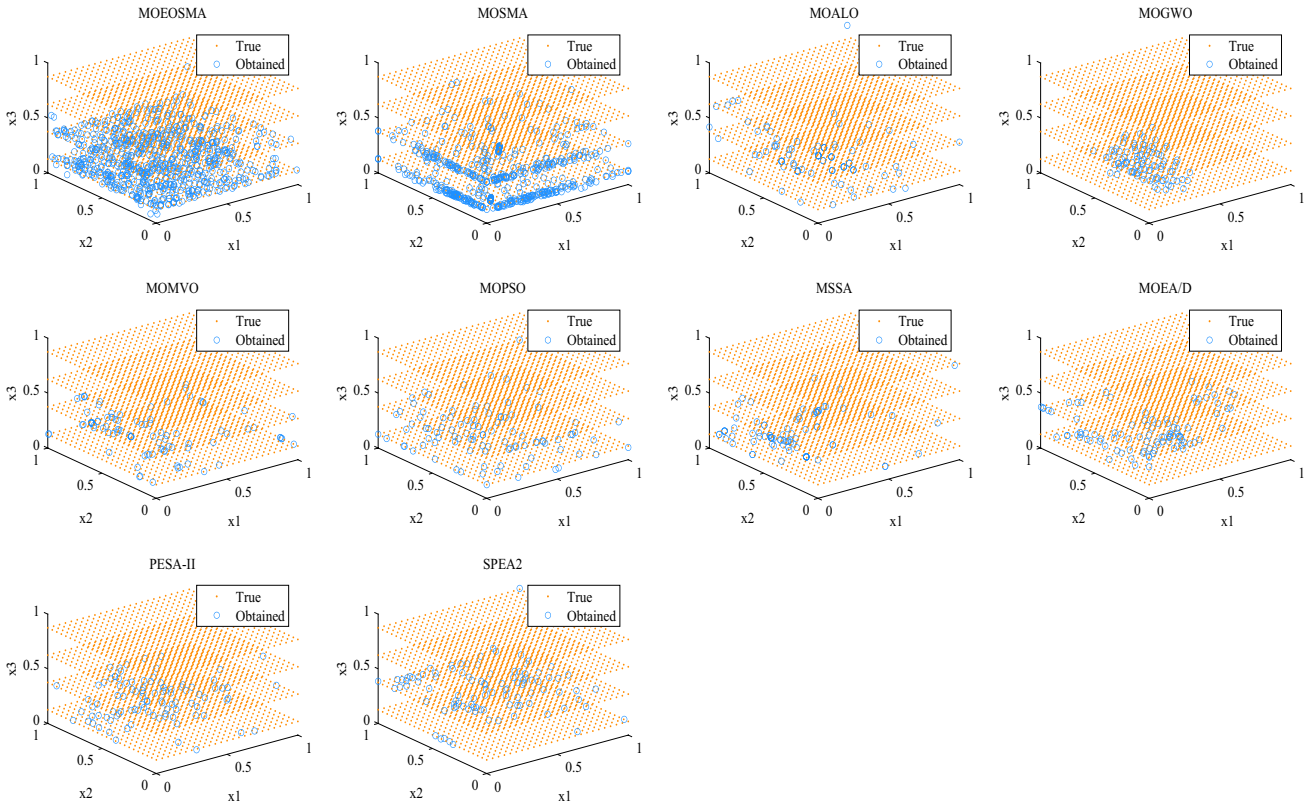


Fig. 5 The optimal PS obtained by all comparison algorithms on MMF16_l3

The obtained PF is close to the true PF and provides the decision-maker with more alternative Pareto optimal solutions. In addition, an interesting phenomenon is that the PS obtained by MOSMA is easily spread near the search space's boundary, while MOEOSMA solves this problem well through the improved boundary updating method. In conclusion, MOEOSMA is highly competitive with other multi-objective algorithms and can achieve better results.

To further illustrate the MOEOSMA's effectiveness and efficiency, Figs. 7 and 8 show the best PS and PF obtained by MOEOSMA on all CEC2020 benchmark functions. As shown in Fig. 7, MOEOSMA can jump out of the local PS, and the obtained PS can cover the true global PS uniformly. It shows that MOEOSMA has a powerful search capability in decision space, which provides a solid platform for solving multimodal multi-objective optimization problems. As shown in Fig. 8, MOEOSMA can approximate the true PF for various types of PF and achieves satisfactory results in terms of convergence, diversity, and uniformity. Overall, MOEOSMA shows superior search performance on CEC2020 benchmark functions. It is verified that the equilibrium pool and the crowding distance can improve the algorithm's convergence accuracy and speed to obtain well-distributed PS and PF.

5 Real-world constrained engineering problems

To test the potential of MOEOSMA, it was applied to eight real-world constraint engineering problems and four large-scale truss optimization problems: speed reducer design, spring design, hydrostatic thrust bearing design, vibrating platform design, car side impact design, water resource management, bulk carriers design, multi-product batch plant, 60-bar truss, 72-bar truss, 200-bar truss, and 942-bar truss optimization problems. These problems contain 2 to 5 objective functions, 3 to 59 decision variables, and 5 to 942 constraints, which can comprehensively analyze the optimization performance of MOEOSMA in addressing various MOPs.

5.1 Real-world optimization problems

The first multi-objective engineering design problem is the speed reducer design problem studied by Kurpati et al. (2002). The objective is to minimize the weight and stress of the reducer. As indicated in Fig. 9, this problem contains seven decision variables: the surface width of the gear (b), the number of pinion teeth (z), the module of teeth (m), the length of the first and second shafts between bearings

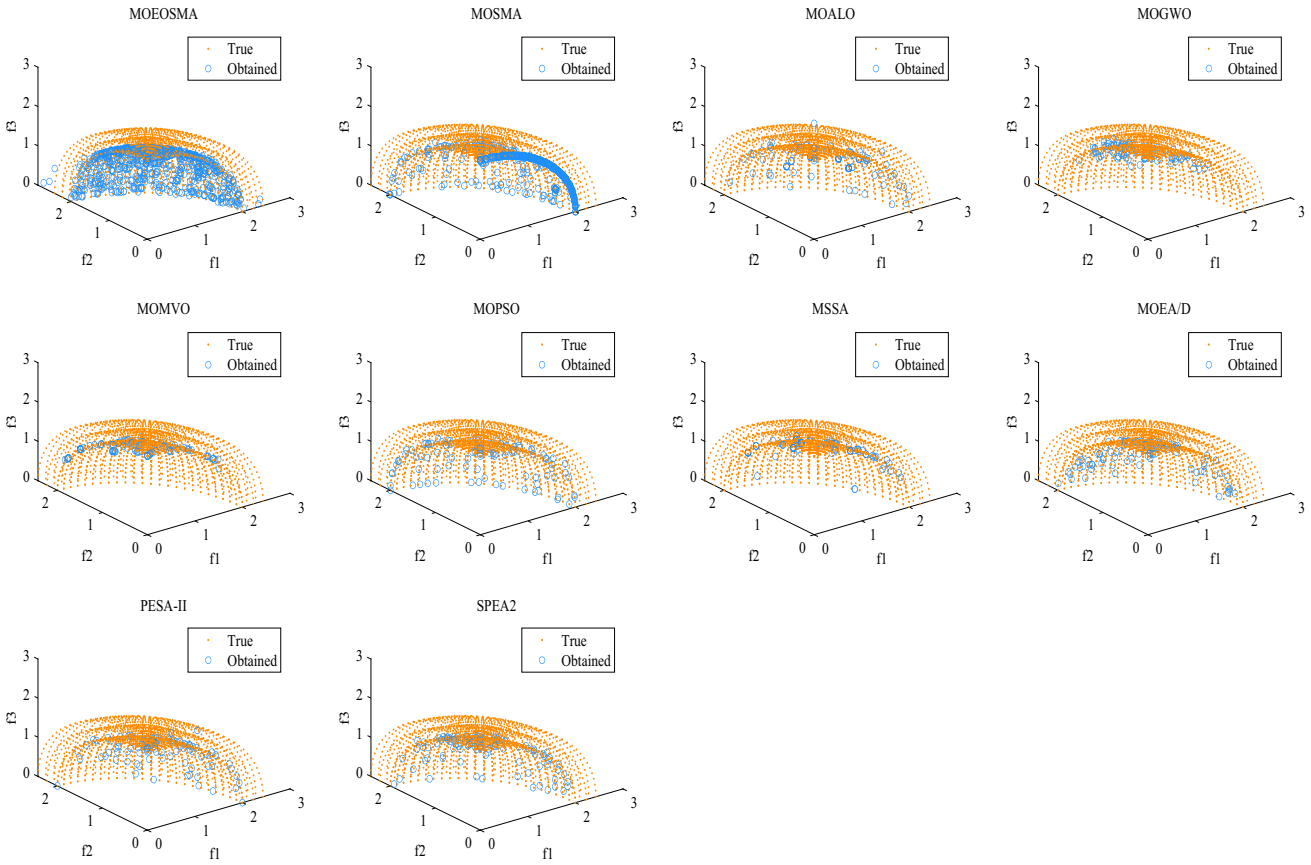


Fig. 6 The optimal PF obtained by all comparison algorithms on MMF16_l3

(l_1, l_2), and the diameter of the first and second shafts (d_1, d_2). The number of pinion teeth (z) is an integer and other variables are continuous. This is a mixed integer problem whose mathematical model is shown in Appendix 1.1 (Dhiman and Kumar 2018).

The second is the spring design problem, as shown in Fig. 10 (Yin et al. 2022a). The objective of this problem is to minimize both stress and volume (Tawhid and Savsani 2019). The design variables are the wire diameter (d), the average coil diameter (D), and the number of active coils (N). The constraints include outside diameter, shear stress, fluctuation frequency and minimum deflection. This problem is unique because all design variables have different characteristics. The number of coil turns can only be taken as an integer, where the wire diameter is standardized and it must be selected from the set of available diameters. The average coil diameter can be considered as a continuous variable. This problem can be formulated as Appendix 1.2.

Third, the objective of the hydrostatic thrust bearing design problem is to minimize the power loss of the hydrostatic thrust bearing during operation while satisfying some constraints (Rao and Savsani 2012; Kumar et al. 2021a). The hydrostatic thrust bearing must bear a specified load when providing axial support. In this study, an objective function is added to minimize the pressure loss of oil inlet and outlet. As shown in Fig. 11, four design variables are considered in this problem: oil viscosity (μ), oil inlet rate (Q), bearing step radius (R), and recess radius (R_o). There are seven constraints related to minimum load carrying capacity, inlet oil pressure requirement, oil temperature increase, oil film thickness, and some physical constraints. It is assumed that all variables are continuous. The mathematical formula for this problem is described in Appendix 1.3.

The fourth problem is a modification of the vibration platform design problem proposed by Messac (1996). It was originally designed as a SOP to maximize the fundamental frequency, with the estimated cost as one of the constraints. Here the problem is modified to include cost as a second objective function and to make the problem combinatorial. Geometry and materials are synthesized in the design process (Narayanan and Azarm 1999). The problem is to design a platform for mounting the motor, as shown in Fig. 12. The setup of the machine is simplified to a pin-pin supported beam bearing the weight. A vibration disturbance is applied from the motor to the beam, which has a length L and a width b and is symmetrical around its middle. Variables d_1

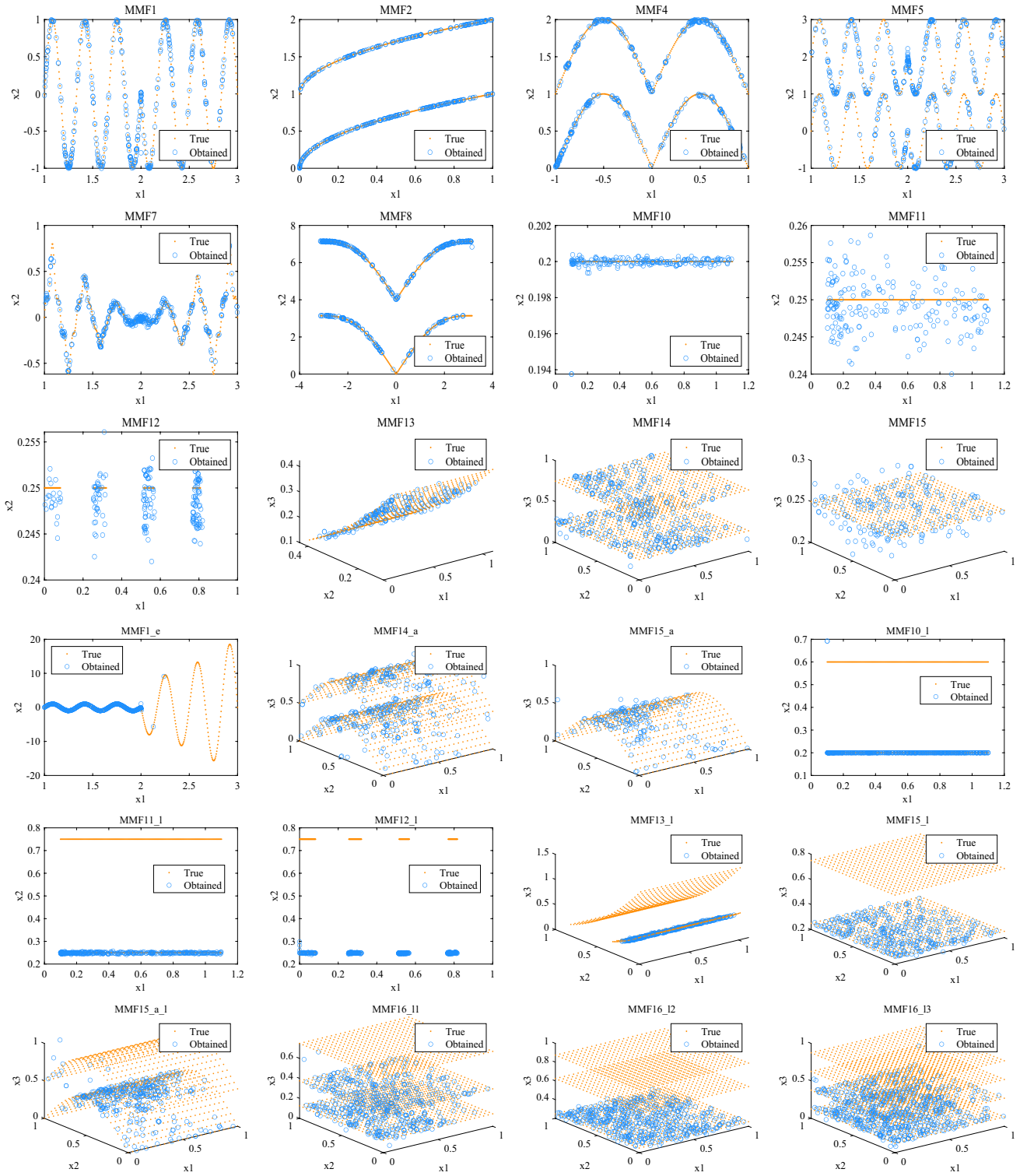


Fig. 7 The optimal PS obtained by MOEOSMA on all CEC2020 multimodal multi-objective benchmark functions

and d_2 locate the contact points of materials 1 and 2 and 2 and 3, respectively. Variable d_3 locates the bottom of the beam. The combined variable M_i refers to the type of material that can form each layer of the beam. The mass density (ρ), Young's modulus of elasticity (E) and cost per unit volume (c) of each material type are shown in Table 4. The objective is to design sandwich beams to minimize the vibration of the beam due to motor disturbance while minimizing

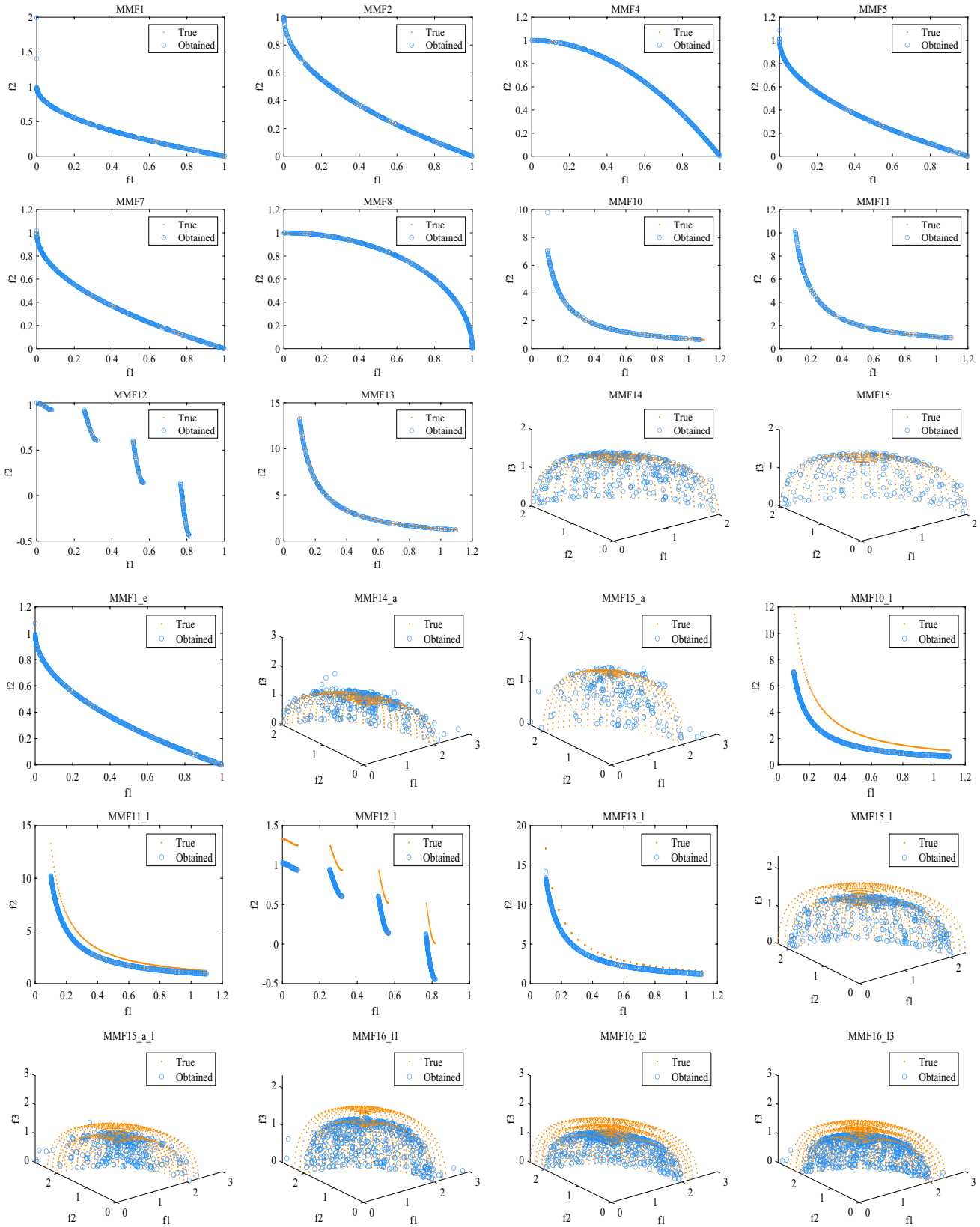


Fig. 8 The optimal PF obtained by MOEOSMA on all CEC2020 multimodal multi-objective benchmark functions

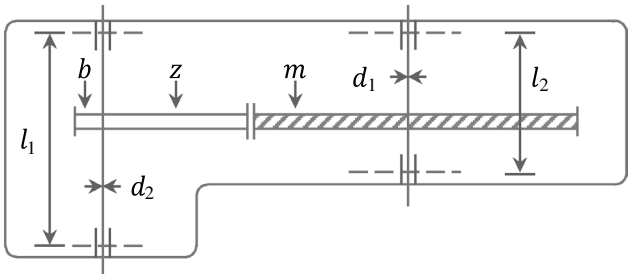


Fig. 9 Schematic diagram of the speed reducer problem

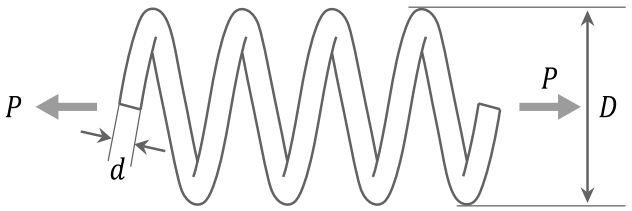


Fig. 10 Schematic diagram of the spring design problem

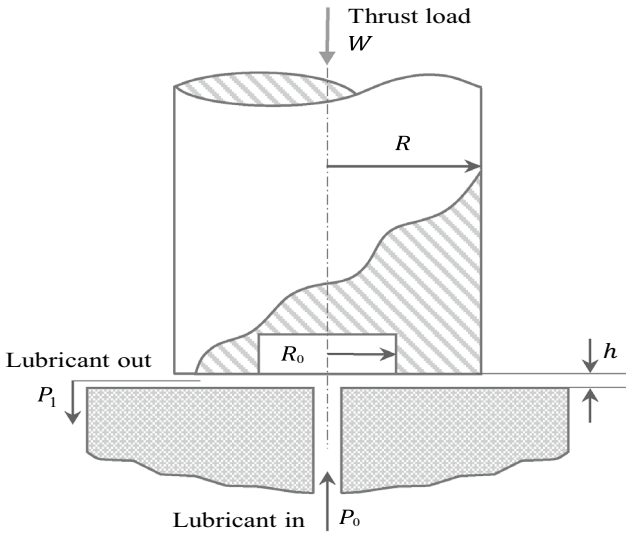


Fig. 11 Schematic diagram of the hydrostatic thrust bearing

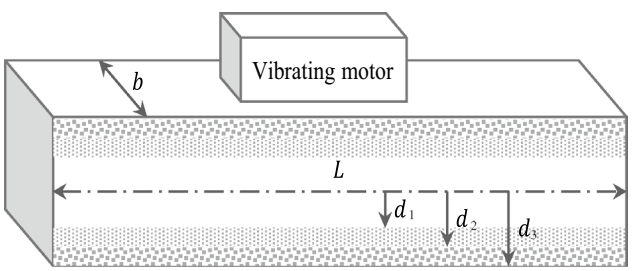


Fig. 12 Schematic diagram of the vibrating platform apparatus

Table 4 Material properties of vibration platform design problem

Material type M_i	ρ (Kg/m^3)	E (N/m^2)	c ($/volume)
1	100	1.6	500
2	2770	70	1500
3	7780	200	800

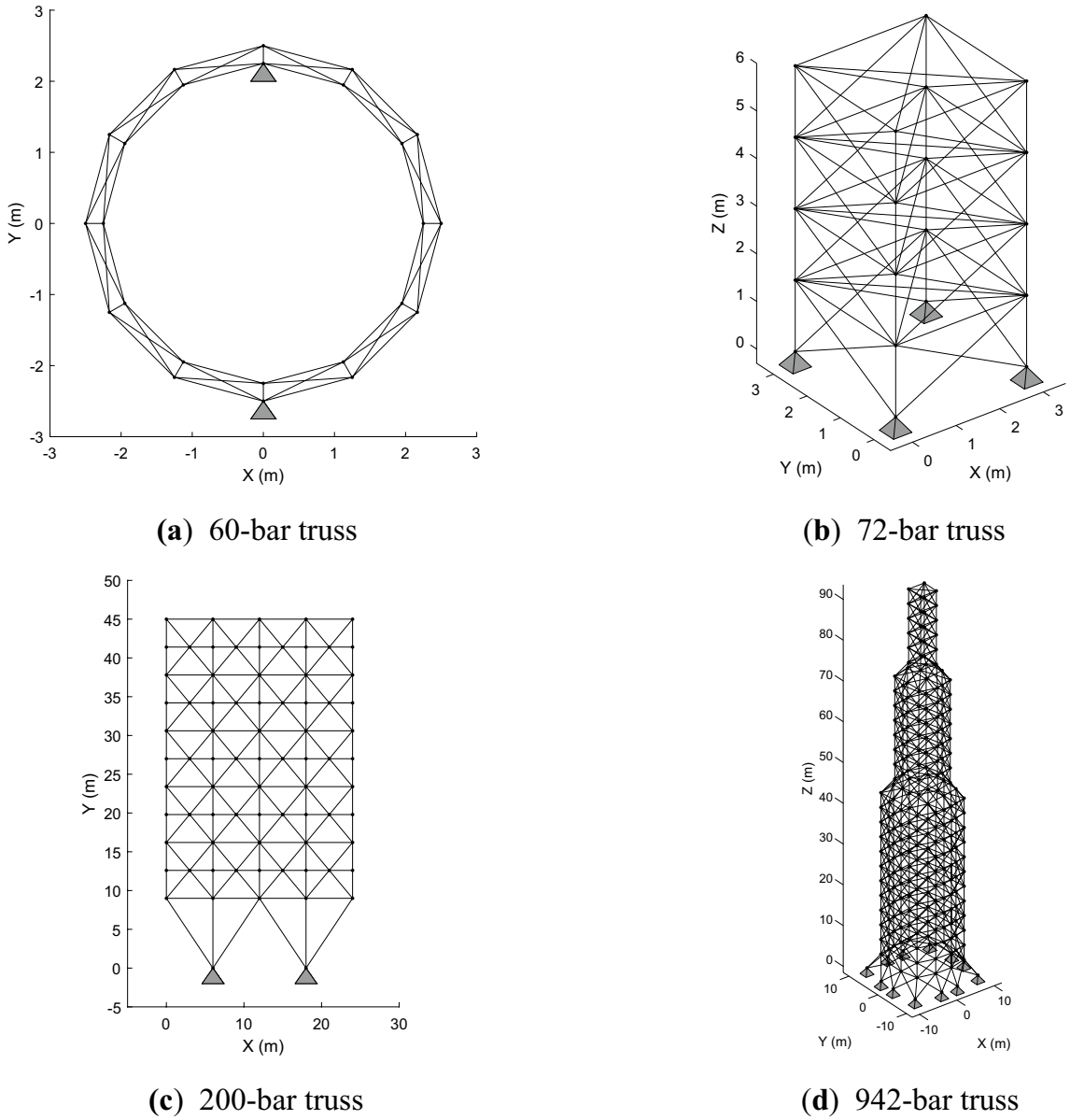
the cost. The complete formulation of the problem is shown in Appendix 1.4.

Fifth, Jain and Deb (2014) developed the car side impact design problem. The objective of this problem is to minimize the weight of the car while minimizing the public forces experienced by the passenger and the average velocity of the V-pillar responsible for withstanding the impact load. All three objectives are in conflict with each other. Therefore, it is expected that there will be a three dimensional trade-off PF. There are ten constraints in this problem, involving limiting values of abdominal load, pubic force, velocity of the V-pillar, rib deflection, etc. There are eleven design variables describing the thicknesses of the B-pillar, floor, crossmembers, door beam, roof rail, etc. Its mathematical description is given in Appendix 1.5.

Sixth, the water resource management is the optimal planning of storm-drainage systems in urban areas, originally proposed by Musselman and Talavage (1980). The formulation of this problem essentially consists of a hierarchically structured linear program with a simulation model as a constraint. It is assumed that there are three decision variables in the drainage system denoting the local detention storage capacity (x_1), maximum treatment rate (x_2) and maximum allowable overflow rate (x_3). The objectives to be optimized are drainage network cost (f_1), storage facility cost (f_2), treatment facility cost (f_3), expected flood damage cost (f_4) and expected flood economic loss (f_5). There are five objective functions for this problem, and the performance of MOEOSMA and other comparison algorithms can be evaluated on the many-objective optimization problem. The mathematical model of this problem is given in Appendix 1.6 (Ray et al. 2001).

Seventh, the bulk carriers design problem is another challenging constraint optimization problem, extracted from (Parsons and Scott 2004). The objectives of the problem are to reduce the transportation cost (f_1), to reduce the weight of the ship (f_2) and to increase the annual cargo volume (f_3). The decision variables of this problem are the length (L), beam (B), depth (D), draft (T), speed (V_k) and block coefficient (C_B) of the ship. The mathematical description of the problem is shown in Appendix 1.7.

Eighth, the multi-product batch plant problem is a complex scheduling problem. The early design of this type of problem is generally to reduce the manufacturing cost and makespan. The multi-product batch plant test problem



(a) 60-bar truss **(b)** 72-bar truss

(c) 200-bar truss **(d)** 942-bar truss

Fig. 13 Schematic diagram of the truss structure. **a** 60-bar truss, **b** 72-bar truss, **c** 200-bar truss, **d** 942-bar truss

is extracted from (Kumar et al. 2021a), which takes into account three objective functions at the same time, with ten decision variables and ten inequality constraints. The mathematical formula for this mixed integer linear programming problem is described in detail in Appendix 1.8.

Finally, four truss optimization problems (60-bar, 72-bar, 200-bar, and 942-bar) are selected from (Pholdee and Bureerat 2013; Tejani et al. 2019; Chou and Truong 2020; Kumar et al. 2021b; Panagant et al. 2021) for validating the performance of MOEOSMA in solving large-scale structural optimization problems. The structural mass and compliance are specified as objective functions subject to allowable stress constraints. The truss optimization problem can be formulated as Eq. (16) (Pholdee and Bureerat 2013; Panagant et al. 2021).

$$\begin{aligned} &\text{Consider } \mathbf{A} = [A_1, A_2, \cdots, A_m] \\ &\text{Minimize } f_1(\mathbf{A}) = \sum_{i=1}^{m} A_i \rho_i L_i \\ &f_2(\mathbf{A}) = \mathbf{u}^{\mathrm{T}}\mathbf{F} \\ &\text{subject to } \left|\sigma_i\right| - \sigma_i^{\max} \le 0 \text{ (stress constraint)} \\ &A_i^{lb} \le A_i \le A_i^{ub} \text{ (side constraint).} \end{aligned} \tag{16}$$

where f_1 denotes the structural mass, f_2 denotes compliance, A_i is the design variable, ρ_i is the density, L_i is the element

Table 5 Characteristics of real-world constraint engineering problems

Problems	Objective	Variable	Constraint
CMOP01: speed reducer design	2	7	11
CMOP02: spring design problem	2	3	8
CMOP03: hydrostatic thrust bearing design	2	4	7
CMOP04: vibrating platform design	2	5	5
CMOP05: car side impact design	3	7	10
CMOP06: water resource management	5	3	7
CMOP07: bulk carriers design	3	6	9
CMOP08: multi-product batch plant design	3	10	10
CMOP09: 60-bar 2D truss optimization	2	25	60
CMOP10: 72-bar 3D truss optimization	2	16	72
CMOP11: 200-bar 2D truss optimization	2	29	200
CMOP12: 942-bar 3D truss optimization	2	59	942

Table 6 Reference points of real-world engineering problems

Problems	References points
CMOP01: speed reducer design	(6.6659849E + 03, 1.2068713E + 03)
CMOP02: spring design problem	(3.0743229E + 01, 2.0764019E + 05)
CMOP03: hydrostatic thrust bearing design	(1.1177181E + 04, 4.5169386E − 06)
CMOP04: vibrating platform design	(− 2.8933091E-03, 7.8120924E + 02)
CMOP05: car side impact design	(4.6224367E + 01, 4.3993211E + 00, 1.3734738E + 01)
CMOP06: water resource management	(8.4580225E + 04, 1.4845700E + 03, 3.1375465E + 05, 9.0794428E + 06, 2.7482920E + 04)
CMOP07: bulk carriers design	(− 8.9553699E + 02, 1.2135542E + 04, 4.7147918E + 03)
CMOP08: multi-product batch plant design	(2.5594278E + 05, 4.9671972E + 04, 6.5576579E + 03)
CMOP09: 60-bar 2D truss optimization	(1.1261826E + 04, 9.1297289E + 04)
CMOP10: 72-bar 3D truss optimization	(3.8673748E + 04, 1.3659625E + 05)
CMOP11: 200-bar 2D truss optimization	(1.5947404E + 05, 2.6859394E + 05)
CMOP12: 942-bar 3D truss optimization	(1.6090635E + 06, 2.3345349E + 05)

length, **u** denotes displacement, **F** denotes loading, σ_i is the element stress, and $\sigma_i^{\max}$ is the maximum stress occurs on the element structure.

The displacement and loading vectors in Eq. (16) are employed from finite element analysis. The material density, modulus of elasticity, and allowable stress are set as 7850 kg/m^3, 200GPa, and 400 MPa, respectively. In practice, the size of each structural member is usually discrete design variable due to beam standard sizing; therefore, the sizing variables are specified as discrete. The structures of the four trusses are shown in Fig. 13, where 60-bar and 200-bar are planar (2D) trusses and 72-bar and 942-bar are spatial (3D) trusses. The number of design variables may not be equal to the number of truss members due to the presence of grouped design variables. The number of design variables for 60-bar, 72-bar, 200-bar, and 942-bar are 25, 16, 29, and 59, respectively. Table 5 depicts the features of these real-world constraint engineering problems.

5.2 Constraint handling method

The penalty function is the most popular approach when dealing with constraints because it provides the unconstrained equivalent of the constraint problem. A good penalty function works in such a way that a feasible solution should have a smaller penalty function value than an infeasible solution. For two specific feasible solutions, the solution with the lower objective function is better. For two infeasible solutions, the less constraint violations the better. Therefore, during the optimization process, a particular penalty value is added to the infeasible solution to guide the search agent away from the infeasible region, as shown in Eq. (17) (Savsani and Savsani 2016).

$$O_n(\mathbf{x}) = f_n(\mathbf{x}) + w \cdot \sum_{i=1}^{l} \max\left(0, g_i(\mathbf{x})\right),\ n = 1, 2, \cdots, M \quad (17)$$

where $O_n(\mathbf{x})$ indicates the nth objective function value, $f_n(\mathbf{x})$ indicates the objective function value without taking

Table 7 The HV values obtained by all comparison algorithms on engineering problems

Algorithms	Index	Speed reducer	Spring design	Hydrostatic thrust bearing	Vibrating platform	Car side impact	Water resource	Bulk carriers	Multi-product batch plant	60-bar truss	72-bar truss	200-bar truss	942-bar truss	FAR (Rank)
MOE-OSMA	Mean	**1.84E+06**	**3.47E+06**	**1.38E−02**	**3.99E+00**	1.37E+01	2.18E+07	4.50E+06	5.45E+09	**5.87E+08**	**3.64E+09**	**3.43E+10**	**3.33E+11**	**2.17 (1)**
	Std.	**1.03E+03**	**2.64E+02**	7.27E−04	**1.55E−02**	7.08E−01	1.36E+06	8.10E+04	**6.02E+07**	**2.14E+06**	**7.21E+06**	**3.18E+07**	**4.68E+08**	**1.83 (1)**
MOSMA (Premkumar et al. 2021b)	Mean	1.68E+06	1.01E+06	8.10E−05	1.40E+00	1.42E+01	2.17E+07	1.34E+05	1.15E+09	5.32E+08	3.05E+09	3.10E+10	3.04E+11	8.67 (10)
	Std.	9.44E+03	9.73E+05	3.62E−04	6.88E−01	**2.59E−01**	3.08E+06	2.51E+05	1.32E+09	1.52E+07	1.33E+08	5.52E+08	6.66E+09	7.75 (7)
MOALO (Mirjalili et al. 2017c)	Mean	1.81E+06	3.31E+06	1.91E−03	2.64E+00	1.15E+01	1.90E+07	3.65E+06	4.48E+09	5.04E+08	3.39E+09	3.26E+10	3.18E+11	8.08 (7.5)
	Std.	2.00E+04	9.19E+04	2.58E−03	7.94E-01	1.20E+00	4.24E+06	5.58E+05	5.45E+08	2.17E+07	8.46E+07	3.92E+08	6.20E+09	9.17 (12)
MOGWO (Mirjalili et al. 2016)	Mean	1.80E+06	3.44E+06	9.90E−03	3.06E+00	1.19E+01	1.36E+07	4.60E+06	5.46E+09	5.81E+08	3.62E+09	3.41E+10	3.32E+11	4.50 (4)
	Std.	1.07E+04	2.18E+04	8.19E−04	5.82E−01	7.23E−01	2.55E+06	9.37E+04	3.26E+08	5.72E+06	1.30E+07	6.36E+07	6.05E+08	4.33 (3)
MOMPA (Zhong et al. 2021)	Mean	1.84E+06	3.46E+06	8.85E−03	3.98E+00	1.36E+01	1.81E+07	4.56E+06	**5.79E+09**	5.61E+08	3.57E+09	3.40E+10	3.24E+11	3.50 (3)
	Std.	2.24E+03	1.82E+03	2.11E−03	3.15E−02	8.93E−01	3.87E+06	7.16E+04	2.73E+08	1.25E+07	5.43E+07	2.42E+08	2.29E+10	5.08 (4)
MOMVO (Mirjalili et al. 2017b)	Mean	1.82E+06	3.42E+06	9.43E−03	3.69E+00	**1.44E+01**	2.17E+07	**4.65E+06**	5.60E+09	5.56E+08	3.61E+09	3.36E+10	3.21E+11	3.33 (2)
	Std.	5.80E+03	1.00E+04	3.46E−03	2.90E−01	1.46E+00	4.36E+06	**6.00E+04**	2.91E+08	5.77E+06	1.62E+07	9.81E+07	9.75E+08	5.33 (5)
MOPSO (Coello et al. 2004)	Mean	5.27E+05	9.61E+05	1.40E−04	4.86E−01	1.41E+01	1.92E+07	2.10E+06	2.17E+09	5.19E+08	3.52E+09	3.35E+10	3.13E+11	8.08 (7.5)
	Std.	7.04E+05	1.07E+06	6.27E−04	5.98E−01	3.96E−01	3.19E+06	1.43E+06	1.04E+09	1.56E+07	4.12E+07	3.81E+08	8.00E+09	8.17 (8)
MSSA (Mirjalili et al. 2017a)	Mean	1.80E+06	3.34E+06	5.69E−03	3.57E+00	1.19E+01	1.80E+07	4.34E+06	5.18E+09	4.32E+08	2.96E+09	2.80E+10	2.66E+11	8.17 (9)
	Std.	1.57E+04	5.09E+04	2.66E−03	2.04E−01	1.40E+00	4.57E+06	1.46E+05	4.76E+08	1.37E+07	1.23E+08	9.96E+08	1.02E+10	8.75 (9)
MODA (Mirjalili 2016)	Mean	1.74E+06	2.00E+06	6.44E−04	2.97E+00	1.17E+01	1.78E+07	3.81E+06	2.75E+09	5.05E+08	3.26E+09	2.84E+10	2.53E+11	9.25 (11)
	Std.	2.30E+04	9.15E+05	1.25E−03	5.99E−01	9.28E−01	3.96E+06	5.70E+05	9.53E+08	1.69E+07	7.40E+07	8.14E+08	7.39E+09	9.00 (11)
MOEA/D (Zhang and Li 2007)	Mean	4.00E+05	8.90E+05	0.00E+00	3.52E−01	1.04E+01	1.56E+07	7.19E+05	2.08E+09	3.72E+08	2.59E+09	2.40E+10	2.22E+11	11.75 (12)
	Std.	4.66E+05	1.04E+06	**0.00E+00**	4.70E−01	1.27E+00	2.68E+06	8.55E+05	1.16E+09	2.24E+07	1.31E+08	8.62E+08	6.12E+09	8.92 (10)

Table 7 (continued)

Algorithms	Index	Speed reducer	Spring design	Hydrostatic thrust bearing	Vibrating platform	Car side impact	Water resource	Bulk carriers	Multi-product batch plant	60-bar truss	72-bar truss	200-bar truss	942-bar truss	FAR (Rank)
PESA-II (Corne et al. 2001)	Mean	1.84E+06	3.40E+06	4.30E − 03	3.57E+00	1.39E+01	**2.34E+07**	4.38E+06	5.37E+09	5.11E+08	3.47E+09	3.28E+10	3.04E+11	5.75 (6)
	Std.	2.47E+03	2.60E+04	2.52E − 03	3.74E − 01	4.74E − 01	**8.89E+04**	1.57E+05	3.28E+08	1.52E+07	4.98E+07	3.76E+08	3.92E+09	5.50 (6)
SPEA2 (Zitzler et al. 2001)	Mean	1.82E+06	3.44E+06	3.75E − 03	3.67E+00	1.41E+01	2.29E+07	4.51E+06	5.09E+09	5.45E+08	3.55E+09	3.31E+10	3.08E+11	4.75 (5)
	Std.	6.38E+03	1.82E+04	2.22E − 03	1.82E − 01	4.64E − 01	2.52E+06	9.42E+04	1.31E+08	7.32E+06	2.74E+07	2.49E+08	3.64E+09	4.17 (2)

*Bold indicates the optimal result, FAR stands for Friedman's average ranking

the constraints into account, $w = 10^8$ represents the static penalty coefficient.

5.3 Performance metrics

Two performance metrics, the Hypervolume (HV) (Panagant et al. 2021) and the Spacing-to-Extent (STE) (Tejani et al. 2019), are used to measure the performance of the optimization algorithm. HV is used to measure the convergence and extension of the PF, while STE is the ratio between spacing and extent of the PF. In this research, the STE value is set to 100 if there is only one solution in the PF; if there are two solutions in the PF, the STE value is set to 10; otherwise, the STE value is calculated according to Eq. (18).

$$\begin{aligned} STE &= Spacing / Extent \\ Spacing &= \frac{1}{|\mathrm{PF}| - 1} \sum_{i=1}^{|\mathrm{PF}|} \left(d_i - \overline{d} \right)^2 \\ Extent &= \sum_{i=1}^{M} \left| f_i^{\max} - f_i^{\min} \right| \end{aligned} \tag{18}$$

where |PF| denotes the number of solutions in the obtained PF, d_i is the Euclidean distance between the objective function vector of the ith solution and its nearest neighbor, $\overline{d}$ is the average of all d_i, M is the number of objective functions, $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of the ith objective function in the PF, respectively.

The superior PF has a larger HV value and a smaller STE value. For the HV metric, the reference point for each test problem is 1.1 times the maximum objective value of the PF obtained by 100 independent runs of MOEOSMA, as shown in Table 6. If all solutions in the PF obtained by the algorithm are dominated by the reference point, the HV value is set to 0.

5.4 Discussion of results

In order to verify the efficiency of the proposed algorithm, the results obtained by MOEOSMA were compared with eleven well-known MOAs, including MOSMA (Premkumar et al. 2021b), MOALO (Mirjalili et al. 2017c), MOGWO (Mirjalili et al. 2016), multi-objective marine predator algorithm (MOMPA) (Zhong et al. 2021), MOMVO (Mirjalili et al. 2017b), MOPSO (Coello et al. 2004), MSSA (Mirjalili et al. 2017a), multi-objective dragonfly algorithm (MODA) (Mirjalili 2016), MOEA/D (Zhang and Li 2007), PESA-II (Corne et al. 2001), SPEA2 (Zitzler et al. 2001). All algorithms have a population size of 100, an archive capacity of 100, and run 30 times independently. The maximum number of iterations is 200 for CMOP01 to CMOP08 and 500 for the four truss optimization problems.

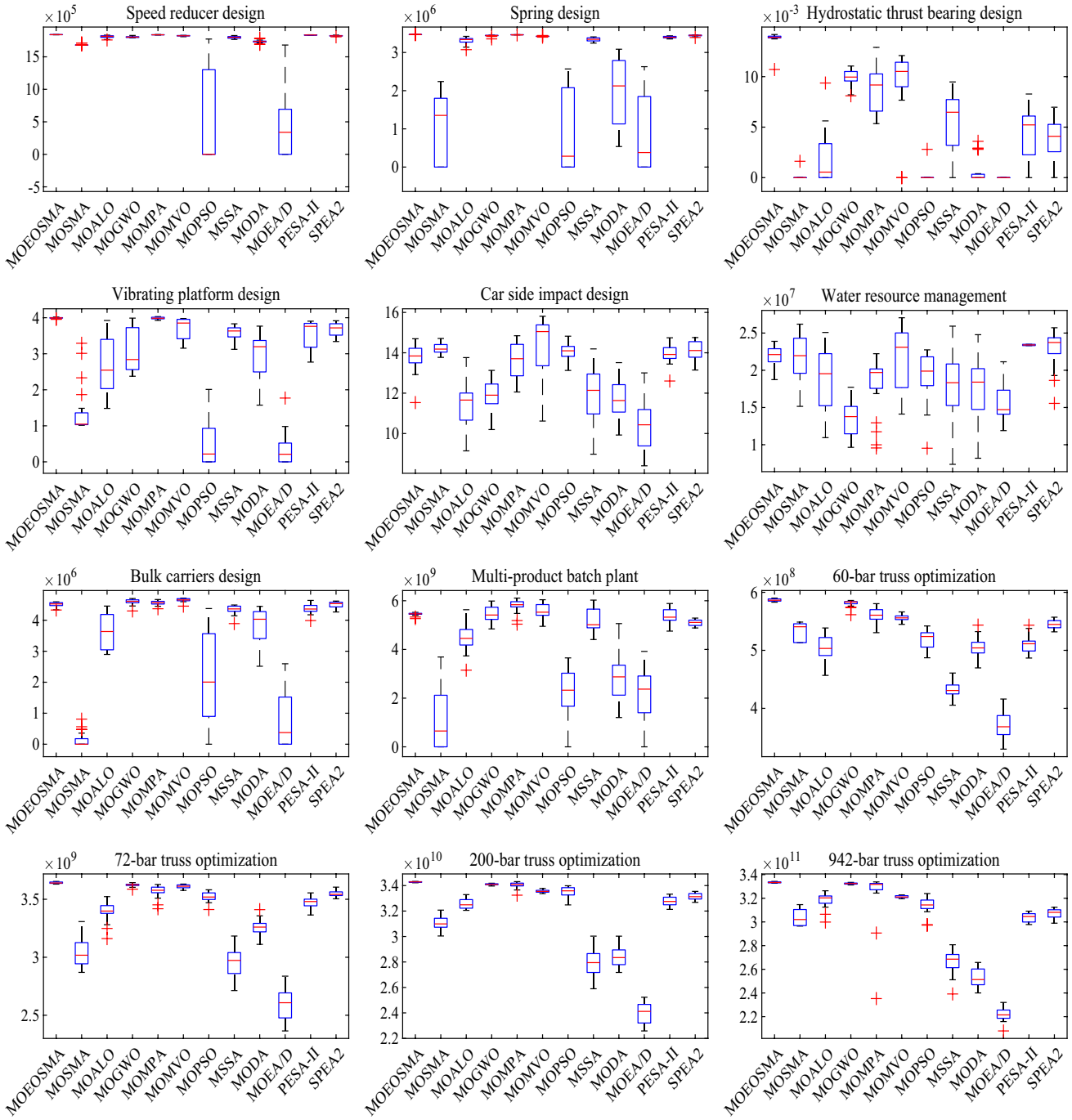


Fig. 14 The box plot of HV values obtained by all comparison algorithms

The mean and standard deviation of HV values obtained by MOEOSMA and other comparison algorithms on twelve engineering problems are presented in Table 7 and Fig. 14. As can be seen from Table 7, MOEOSMA, MOMPA, MOMVO, and PESA-II obtained the best HV values on 8, 1, 2, and 1 engineering problems, respectively. It is worth noting that MOEOSMA outperforms other algorithms on engineering problems with two objective functions, but the performance in solving engineering problems with more than two objectives needs to be improved. This is because MOEOSMA is updated based on the elite Pareto optimal solution in the equilibrium pool. When solving many-objective optimization problems, the number of non-dominated solutions increases exponentially. Due to the low selection pressure caused by the Pareto dominance relationship, it is difficult for MOEOSMA to select the elite non-dominated

Table 8 The STE values obtained by all comparison algorithms on engineering problems

Algorithms	Index	Speed reducer	Spring design	Hydrostatic thrust bearing	Vibrating platform	Car side impact	Water resource	Bulk carriers	Multi-product batch plant	60-bar truss	72-bar truss	200-bar truss	942-bar truss	FAR (Rank)
MOE-OSMA	Mean	0.007616	**0.00763**	**0.006002**	0.006363	0.01163	0.007966	0.009807	0.009202	0.006861	0.0064	0.005754	0.006306	3.08 (2)
	Std.	0.001619	**0.000608**	0.006687	0.002779	0.002504	0.002249	0.002461	0.003186	0.001602	**0.001318**	0.002663	**0.00104**	**2.75 (1)**
MOSMA (Premkumar et al. 2021b)	Mean	6.190781	63.03975	100	0.120528	0.015876	0.007693	43.58403	82	0.019116	0.026808	0.023481	0.009425	9.58 (11)
	Std.	4.787493	46.52118	**0**	0.090103	0.002872	0.002505	47.41154	36.93522	0.005867	0.009442	0.008318	0.002014	8.08 (10)
MOALO (Mirjalili et al. 2017c)	Mean	0.008389	0.016724	10.00799	0.010147	0.018728	0.01985	0.012644	0.011457	0.012697	0.010234	0.010462	0.007767	7.67 (9)
	Std.	0.002179	0.011274	30.77662	0.008675	0.005844	0.01241	0.004584	0.007989	0.006532	0.005392	0.005172	0.002564	7.42 (8)
MOGWO (Mirjalili et al. 2016)	Mean	0.01016	0.009782	0.052808	0.027798	0.011509	0.015329	0.013199	0.014172	0.008917	0.008246	0.006471	0.008827	6.67 (7)
	Std.	0.002279	0.002428	0.019101	0.014908	0.002978	0.005729	0.008092	0.006158	0.002702	0.001957	0.000913	0.001522	5.33 (5)
MOMPA (Zhong et al. 2021)	Mean	0.006831	0.008756	0.006735	0.008062	0.009371	0.007958	0.00994	0.009582	0.007449	0.00657	0.007049	0.00655	3.75 (3)
	Std.	**0.000885**	0.001292	0.008912	0.005284	0.001258	0.002026	0.003714	0.006294	0.005307	0.00597	0.007989	0.007818	4.58 (3.5)
MOMVO (Mirjalili et al. 2017b)	Mean	0.008069	0.010506	10.01727	0.006711	0.015751	0.014813	0.007728	**0.007412**	0.005918	0.007075	0.006168	0.007128	4.67 (4)
	Std.	0.002225	0.00233	30.77346	**0.001962**	0.003699	0.014813	0.002828	0.002211	**0.000923**	0.002056	0.0009	0.001935	4.58 (3.5)
MOPSO (Coello et al. 2004)	Mean	15.51076	86.5	85.01315	75.50934	0.008634	0.007823	5.018758	100	0.007782	0.006505	0.005436	0.007278	7.33 (8)
	Std.	36.48239	32.97128	36.60266	43.56894	**0.000615**	**0.001944**	22.35627	**0**	0.002177	0.001903	**0.000693**	0.001406	5.50 (6)
MSSA (Mirjalili et al. 2017a)	Mean	0.005203	0.023133	5.003872	**0.00589**	0.018678	0.014917	**0.006726**	0.009733	0.012543	0.025977	0.019935	0.015556	6.42 (6)
	Std.	0.002594	0.011888	22.35977	0.002092	0.004933	0.008638	0.002319	0.006637	0.010824	0.021056	0.010965	0.009545	7.58 (9)
MODA (Mirjalili 2016)	Mean	0.018683	51.03311	35.00655	0.017284	0.027097	0.010567	0.022568	0.030992	0.03454	0.023413	0.028839	0.03324	9.42 (10)
	Std..	0.008528	50.32135	48.93112	0.01076	0.007679	0.004597	0.018947	0.034422	0.022554	0.019054	0.022925	0.029947	10.08 (11)
MOEA/D (Zhang and Li 2007)	Mean	91	72.02123	95.5	73	0.062938	0.08805	24.03191	67.02007	0.159011	0.086537	0.097794	1.1894	11.50 (12)
	Std.	27.70142	43.92514	20.12461	42.31461	0.034481	0.049655	39.23574	46.20678	0.100285	0.056819	0.053277	3.015823	11.17 (12)
PESA-II (Corne et al. 2001)	Mean	0.00587	0.013163	0.01353	0.008845	0.010596	0.007578	0.010543	0.0127	0.010353	0.009122	0.007997	0.009438	5.50 (5)
	Std.	0.003673	0.005469	0.015186	0.003158	0.001681	0.00222	0.00969	0.010836	0.011838	0.006936	0.00416	0.008746	6.50 (7)

Table 8 (continued)

Algorithms	Index	Speed reducer	Spring design	Hydrostatic thrust bearing	Vibrating platform	Car side impact	Water resource	Bulk carriers	Multi-product batch plant	60-bar truss	72-bar truss	200-bar truss	942-bar truss	FAR (Rank)
SPEA2 (Zitzler et al. 2001)	Mean	**0.0047**	0.009576	0.088184	0.011765	**0.008008**	**0.007547**	0.006792	0.011049	**0.004764**	**0.006291**	**0.005345**	**0.005973**	**2.42 (1)**
	Std.	0.000975	0.002938	0.12941	0.005599	0.001102	0.003947	**0.00165**	0.013364	0.001658	0.003781	0.001369	0.001552	4.42 (2)

*Bold indicates the optimal result, FAR stands for Friedman's average ranking

solutions. For the three-objective engineering problem, MOMPA and MOMVO achieved better results, and for the five-objective engineering problem, PESA-II achieved the best results, but PESA-II runs 1000 times slower than MOEOSMA. Although MOEOSMA does not perform best on problems with more than two objectives, it still has a strong competitive advantage. In addition, MOEOSMA performs better than other algorithms on large-scale truss optimization problems. According to the NFL theorem (Wolpert and Macready 1997), no algorithm performs best on all problems, and MOEOSMA is more suitable for solving real-world engineering problems with two objectives. In addition, Friedman test results show that MOEOSMA, MOMPA, and MOMVO are superior to other competitive algorithms in terms of convergence and diversity of the PF.

For the HV metric, a larger value indicates better convergence and coverage of the PF. As can be seen from Fig. 14, the minimum HV value obtained by the algorithm is 0, indicating that all solutions obtained by the algorithm are dominated by the reference point of the problem, and this PF is the worst. In addition, MOEOSMA has the highest box plot with the least number of outliers and is narrowest, indicating that the algorithm has good generalization ability and stability. The performance of MOEOSMA is superior to the current MOSMA except for the car side impact problem, which verifies the effectiveness of the improved strategy used in this study. Although MOEOSMA does not obtain the best results for the many-objective optimization problems, it still remains at the same level as several state-of-the-art algorithms.

The STE values of PF produced by MOEOSMA and other comparison algorithms are recorded in Table 8 and Fig. 15. As shown in Table 8, MOEOSMA, MOMVO, MSSA, and SPEA2 obtain the most uniformly distributed PF on 2, 1, 2, and 7 engineering problems, respectively. The Friedman statistical test results show that the overall distribution of PF obtained by SPEA2 is the best, followed by MOEOSMA. The effectiveness and efficiency of the equilibrium pool strategy and crowding distance method on various MOPs are verified. In addition, the Friedman rankings in Tables 7 and 8 show that MODA, MOSMA and MOEA/D do not solve these engineering problems well. The search efficiency of MODA and MOSMA needs to be improved, while MOEA/D may not be good at handling engineering problems with constraints.

For the STE metric, smaller values indicate better uniformity and extensiveness of the PF. As can be seen from Fig. 15, the algorithm obtains a maximum STE value of 100. This situation is because the algorithm obtains a PF with only one solution and cannot calculate the Spacing. Since such a PF is the worst, it is set to a relatively large value. If the obtained PF has only two solutions and also cannot calculate the Spacing, set its STE value to 10. For PF with more than two solutions, the STE value (usually less than 1)

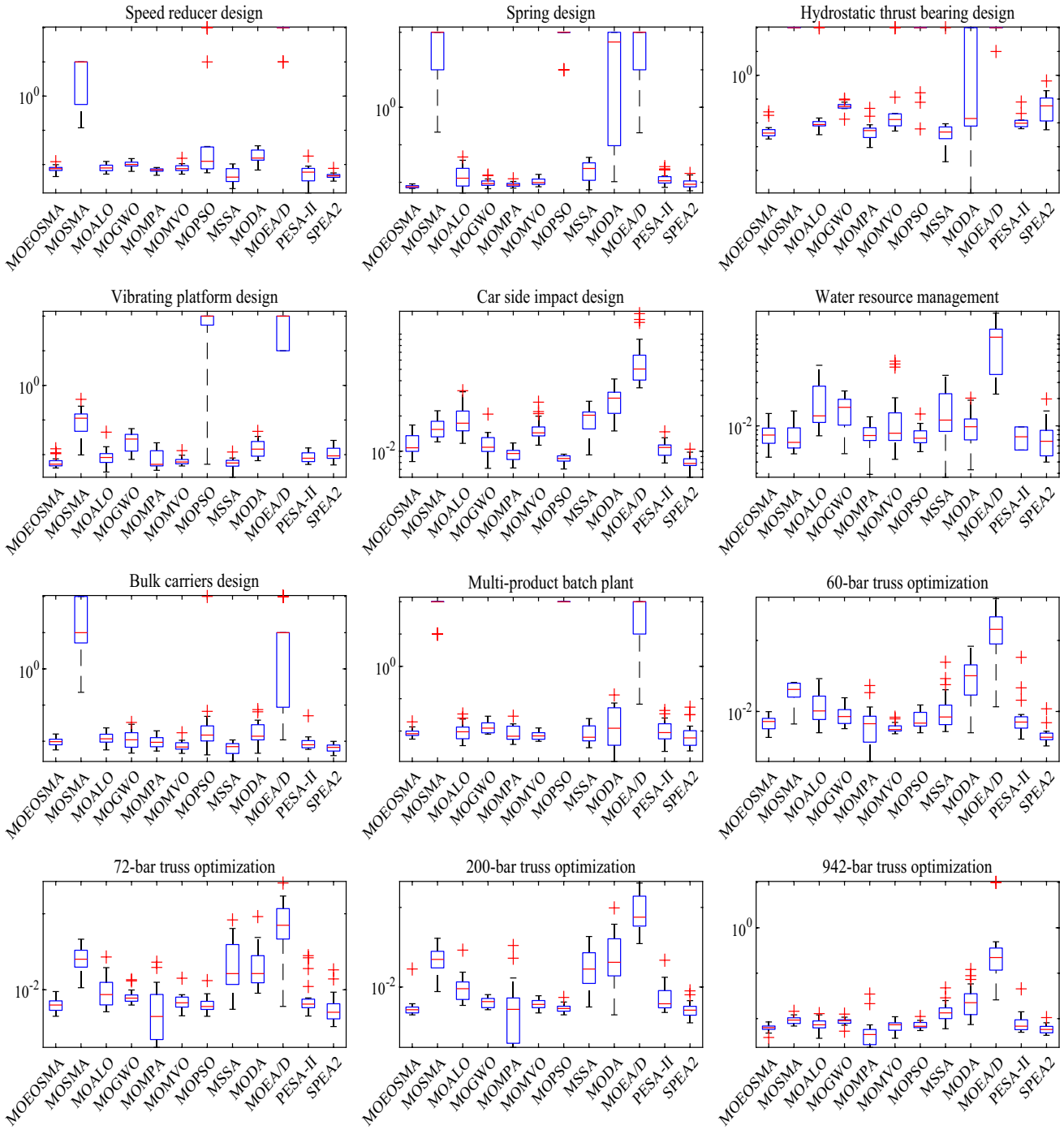


Fig. 15 The box plot of STE values obtained by all comparison algorithms

is calculated by Eq. (18). According to Fig. 15, most algorithms can obtain well-distributed PF. However, MOSMA, MOPSO, MODA, and MOEA/D perform poorly on some problems and can only obtain one or two non-dominated solutions. In addition, SPEA2 obtains the best PF distribution in many problems, but its convergence accuracy is not as good as MOEOSMA.

To avoid the influence of randomness, the Wilcoxon rank-sum test was employed to verify whether the HV and STE values obtained by the paired algorithms are significantly different. Tables 9 and 10 show the results of the paired sample Wilcoxon rank-sum test for MOEOSMA and the other comparison algorithms. Table 9 reveals that MOEOSMA significantly outperforms other algorithms for most

Table 9 Wilcoxon *p*-value test results for the HV metrics (two-tailed)

Paired algorithms	CMOP01	CMOP02	CMOP03	CMOP04	CMOP05	CMOP06	CMOP07	CMOP08	CMOP09	CMOP10	CMOP11	CMOP12
MOEOSMA vs MOSMA	4.49E − 08	5.69E − 08	1.13E − 08	6.80E − 08	8.35E − 03	**8.82E − 01**	2.96E − 08	6.03E − 08	6.76E − 08	6.80E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs MOALO	6.80E − 08	6.80E − 08	6.03E − 08	6.80E − 08	3.07E − 06	2.23E − 02	1.92E − 07	1.20E − 06	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs MOGWO	6.80E − 08	6.80E − 08	1.06E − 07	1.43E − 07	6.01E − 07	6.80E − 08	2.47E − 04	**9.46E − 01**	1.58E − 06	4.54E − 06	6.80E − 08	5.17E − 06
MOEOSMA vs MOMPA	4.70E − 03	6.80E − 08	1.23E − 07	**5.25E − 01**	**7.76E − 01**	4.68E − 05	2.75E − 02	3.29E − 05	6.80E − 08	6.80E − 08	5.87E − 06	9.13E − 07
MOEOSMA vs MOMVO	6.80E − 08	6.80E − 08	2.96E − 07	2.56E − 07	2.94E − 02	**4.73E − 01**	6.92E − 07	**1.72E − 01**	6.80E − 08	7.90E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs MOPSO	4.49E − 08	5.37E − 08	1.13E − 08	5.73E − 08	**1.26E − 01**	1.23E − 03	1.06E − 07	6.79E − 08	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs MSSA	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08	2.60E − 05	3.06E − 03	6.61E − 05	3.60E − 02	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs MODA	6.80E − 08	6.79E − 08	3.48E − 08	6.80E − 08	6.01E − 07	1.29E − 04	2.22E − 07	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs MOEA/D	5.73E − 08	5.37E − 08	8.01E − 09	5.73E − 08	1.43E − 07	2.96E − 07	6.03E − 08	6.79E − 08	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs PESA-II	7.90E − 08	6.80E − 08	6.80E − 08	6.80E − 08	**3.65E − 01**	8.12E − 07	4.70E − 03	2.75E − 02	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08
MOEOSMA vs SPEA2	6.80E − 08	6.80E − 08	6.79E − 08	6.80E − 08	**7.20E − 02**	4.32E − 03	**6.95E − 01**	9.17E − 08	6.80E − 08	6.80E − 08	6.80E − 08	6.80E − 08

*Bold indicates that there is no significant difference between the two algorithms

Table 10 Wilcoxon *p*-value test results for the STE metrics (two-tailed)

Paired algorithms	CMOP01	CMOP02	CMOP03	CMOP04	CMOP05	CMOP06	CMOP07	CMOP08	CMOP09	CMOP10	CMOP11	CMOP12
MOEOSMA vs MOSMA	4.49E − 08	4.26E − 08	8.01E − 09	1.06E − 07	6.61E − 05	**4.90E − 01**	5.56E − 08	2.40E − 08	4.52E − 07	6.80E − 08	9.17E − 08	1.05E − 06
MOEOSMA vs MOALO	**3.79E − 01**	9.21E − 04	1.04E − 04	1.67E − 02	9.75E − 06	1.41E − 05	4.11E − 02	**9.68E − 01**	3.38E − 04	2.34E − 03	1.58E − 06	3.60E − 02
MOEOSMA vs MOGWO	5.63E − 04	3.75E − 04	9.17E − 08	6.01E − 07	**9.25E − 01**	8.29E − 05	**4.41E − 01**	1.78E − 03	9.79E − 03	9.21E − 04	3.05E − 04	4.54E − 06
MOEOSMA vs MOMPA	**1.02E − 01**	1.78E − 03	**7.56E − 01**	**8.82E − 01**	2.34E − 03	**7.97E − 01**	**7.76E − 01**	**2.98E − 01**	**5.98E − 01**	**1.90E − 01**	**7.97E − 01**	3.06E − 03
MOEOSMA vs MOMVO	**7.97E − 01**	3.50E − 06	9.74E − 06	**8.59E − 02**	2.47E − 04	**2.29E − 01**	4.32E − 03	3.15E − 02	4.11E − 02	**2.50E − 01**	2.14E − 03	**1.33E − 01**
MOEOSMA vs MOPSO	7.70E − 03	1.94E − 08	3.18E − 08	3.94E − 07	5.17E − 06	**9.03E − 01**	7.11E − 03	8.01E − 09	**2.85E − 01**	**7.15E − 01**	**3.94E − 01**	2.75E − 02
MOEOSMA vs MSSA	5.12E − 03	1.38E − 06	**6.75E − 01**	**7.56E − 01**	4.68E − 05	1.48E − 03	6.87E − 04	**1.99E − 01**	1.67E − 02	8.60E − 06	5.23E − 07	2.06E − 06
MOEOSMA vs MODA	9.13E − 07	5.36E − 08	4.92E − 04	3.50E − 06	1.66E − 07	4.39E − 02	2.56E − 03	**2.98E − 01**	2.69E − 06	9.17E − 08	3.99E − 06	1.06E − 07
MOEOSMA vs MOEA/D	1.51E − 08	3.43E − 08	1.13E − 08	3.30E − 08	6.80E − 08	6.80E − 08	1.46E − 07	3.94E − 08	6.80E − 08	3.42E − 07	6.80E − 08	6.79E − 08
MOEOSMA vs PESA-II	9.79E − 03	6.92E − 07	1.41E − 05	1.35E − 03	**3.94E − 01**	**5.92E − 01**	**1.48E − 01**	**7.56E − 01**	**7.97E − 01**	**4.90E − 01**	2.34E − 03	**1.56E − 01**
MOEOSMA vs SPEA2	1.80E − 06	8.35E − 03	7.95E − 07	9.28E − 05	1.05E − 06	**1.99E − 01**	8.29E − 05	**7.64E − 02**	9.28E − 05	4.39E − 02	**8.82E − 01**	**1.64E − 01**

*Bold indicates that there is no significant difference between the two algorithms

Table 11 The average number of Pareto solutions obtained by all comparison algorithms

Algorithms	CMOP01	CMOP02	CMOP03	CMOP04	CMOP05	CMOP06	CMOP07	CMOP08	CMOP09	CMOP10	CMOP11	CMOP12
MOEOSMA	**100.00**	**65.75**	**99.10**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
MOSMA (Premkumar et al. 2021b)	2.50	1.55	1.00	10.85	93.65	99.95	1.85	1.20	68.40	28.05	30.30	44.65
MOALO (Mirjalili et al. 2017c)	99.95	38.95	90.10	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	81.75	87.35	85.40	89.25
MOGWO (Mirjalili et al. 2016)	76.70	54.45	34.15	87.40	99.50	99.95	94.15	96.65	92.95	89.10	95.55	97.65
MOMPA (Zhong et al. 2021)	**100.00**	61.95	96.65	**100.00**	**100.00**	**100.00**	**100.00**	99.85	86.75	89.80	85.05	80.55
MOMVO (Mirjalili et al. 2017b)	83.80	45.70	41.90	99.95	**100.00**	**100.00**	**100.00**	**100.00**	99.05	97.65	98.20	99.35
MOPSO (Coello et al. 2004)	43.10	1.15	6.20	12.70	**100.00**	99.95	44.15	1.00	80.75	94.25	95.55	85.65
MSSA (Mirjalili et al. 2017a)	**100.00**	33.30	95.05	**100.00**	**100.00**	**100.00**	**100.00**	99.95	72.65	51.15	50.50	81.80
MODA (Mirjalili 2016)	64.60	3.85	62.05	91.15	**100.00**	99.35	93.85	60.85	52.65	59.80	54.40	38.75
MOEA/D (Zhang and Li 2007)	1.10	1.40	1.05	1.30	11.30	22.85	2.40	1.70	5.65	7.45	7.05	4.30
PESA-II (Corne et al. 2001)	98.20	43.05	66.60	73.50	98.40	99.50	87.40	73.70	78.00	78.05	81.75	79.50
SPEA2 (Zitzler et al. 2001)	**100.00**	55.90	45.90	94.30	**100.00**	**100.00**	**100.00**	99.95	**100.00**	**100.00**	**100.00**	**100.00**

*Bold indicates the optimal result

optimization problems and outperforms all comparison algorithms for speed reducer, spring design, hydrostatic thrust bearing, and four large-scale truss optimization problems. It is verified that the comprehensive performance of MOEOSMA is better than other comparison algorithms, and there are significant differences.

Table 10 illustrates that in terms of uniformity and extensiveness, the PF obtained by MOEOSMA is not significantly different from MOMPA, MOMVO, and PESA-II on 9, 5, and 7 problems, respectively. These algorithms all obtained well-distributed PF. However, there are significant differences between MOEOSMA and MOSMA on eleven problems. As can be seen from Table 8, the STE values of MOEOSMA are smaller than that of MOSMA, indicating that MOEOSMA is significantly better than MOSMA in terms of uniformity. In addition, MOEOSMA is significantly different from MOALO, MOGWO, MSSA, MODA, and MOEA/D. The former obtains better PF distribution, which verifies the performance of MOEOSMA. When the primary search operator of MOAs has sufficient exploration capability, the distribution of PF obtained using the elite archiving mechanism based on the crowding distance (similar to MOPSO) is more uniform than that of the non-dominated ranking (similar to NSGA-II). For high-dimensional complex PF, the distribution does not deteriorate significantly. In contrast, the archiving mechanism based on the non-dominated ranking is more suitable to combine with the search operator with strong exploitation capability, thus improving the exploration capability of MOAs.

In MOPs, the number of non-dominated solutions in the PS obtained by the algorithm is crucial. It is detrimental for the user to weigh the decisions if the number of solutions is too small. Therefore, the number of Pareto optimal solutions obtained can be regarded as a diversity indicator. Table 11 statistics the average number of Pareto optimal solutions obtained by all comparison algorithms run 30 times independently on each problem. The best results are shown in bold. Because the archive capacity is set to 100 for all algorithms, the average number of Pareto optimal solutions in Table 11 is at most 100. According to Table 11, MOEOSMA obtains the most non-dominated solutions on twelve engineering problems, while other algorithms can only obtain more non-dominated solutions on some of the problems. It is demonstrated that using MOEOSMA to solve real-world MOPs is more beneficial for users to weigh and select the most satisfactory solution among multiple objective functions.

The quality measure for PF is very complicated, and there is no unary quality measure that can indicate that approximate set A is superior to B (Zitzler et al. 2003). Because the theoretical PF is usually unknown in real-world optimization problems, it is challenging to design a satisfactory binary quality measure. Such a metric also has the disadvantage that the evaluation is not objective. A direct comparison of PF

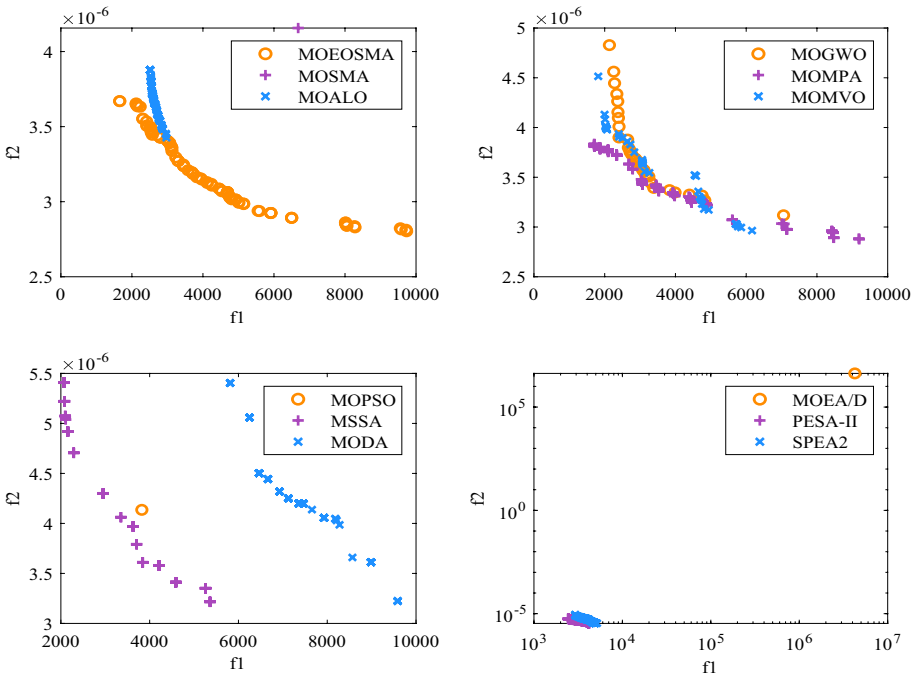


Fig. 16 The optimal PF obtained by all comparison algorithms on the hydrostatic thrust bearing design problem

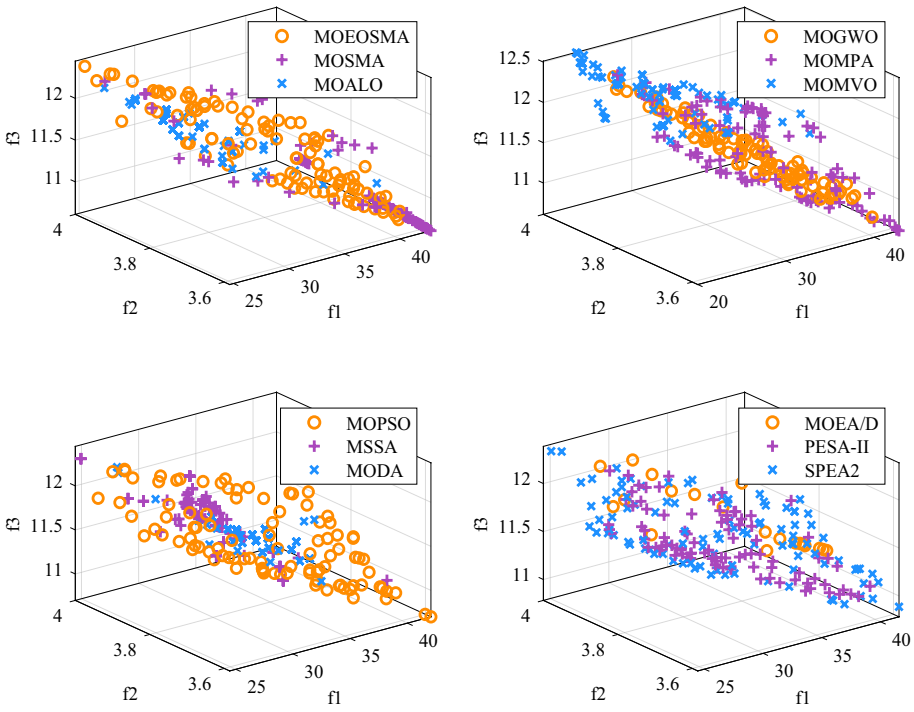


Fig. 17 The optimal PF obtained by all comparison algorithms on the car side impact design problem

can more accurately demonstrate the advantages of different algorithms. Figures 15, 16, 17, and 18 present the optimal PF obtained by all comparison algorithms for the hydrostatic thrust bearing design, car side impact design, water resource management, and 60-bar truss optimization problem, respectively. The PF for other engineering problems is shown in Appendix Figs. 20, 21, 22, 23, 24, 25, 26, and 27.

As shown in Fig. 16, the PF obtained by MOALO with MOSMA is strictly dominated by the PF obtained by MOEOSMA on the hydrostatic thrust bearing design problem. MOSMA and MOPSO can only discover a few non-dominated solutions that satisfy all constraints, whereas MOEA/D cannot find feasible solutions. It means that the search operators of these three algorithms are inefficient at solving the

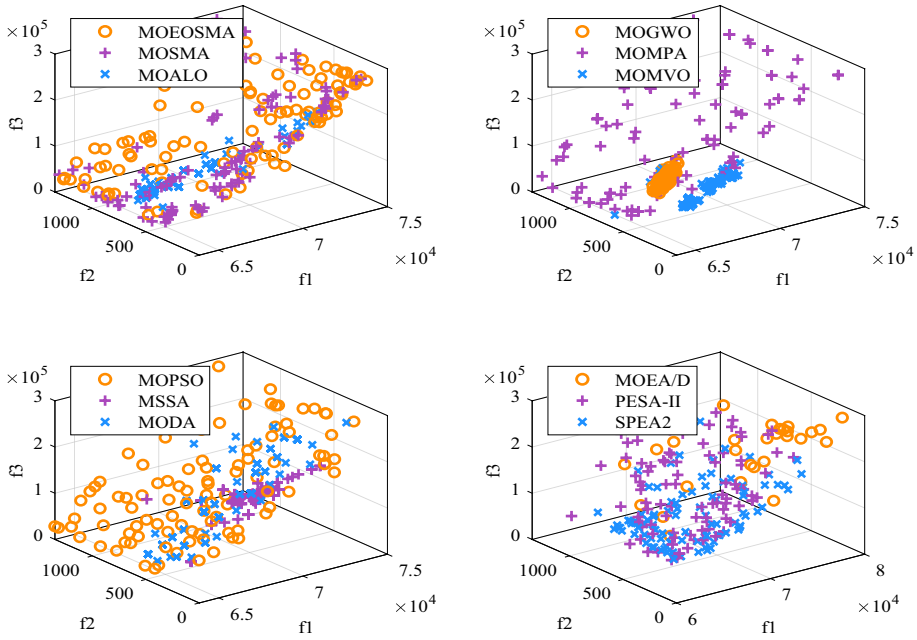


Fig. 18 The optimal PF obtained by all comparison algorithms on the water resource management problem

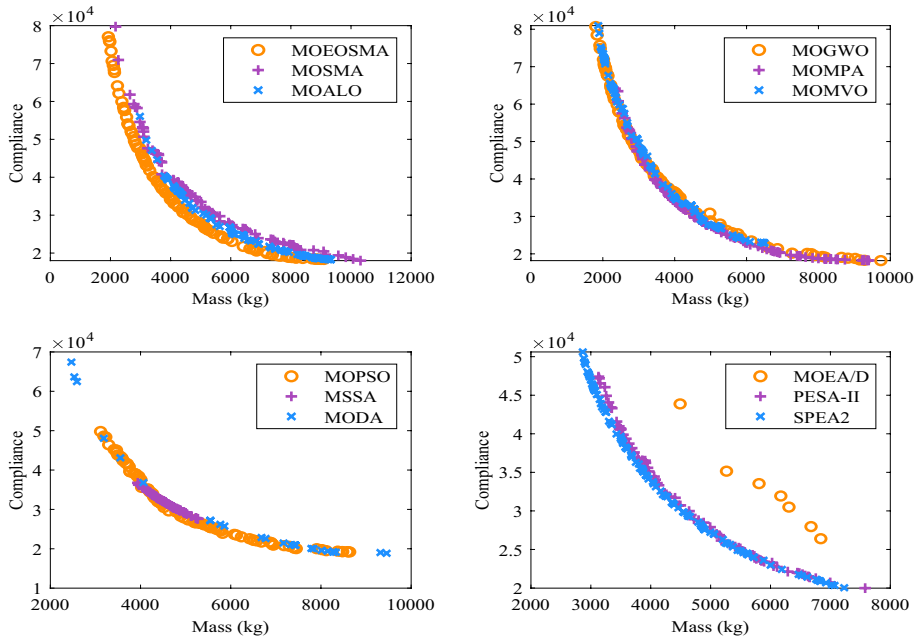


Fig. 19 The optimal PF obtained by all comparison algorithms on the 60-bar truss optimization problem

problem. Compared with other algorithms, MOEOSMA obtains a more uniform and extensive PF.

The optimal PF of the car side impact design problem, depicted in Fig. 17, illustrates the search performance of MOEOSMA in comparison to other algorithms on the three objective optimization problems. In this instance, the HV value indicates that MOMVO obtained the best convergence and diversity of PF, but its distribution is not the most uniform, with many non-dominated solutions concentrated in the same region. As can be seen from Fig. 17, MOPSO, MOMPA, and SPEA2 are the three algorithms with the best performance, followed by MOEOSMA and PESA-II.

Figure 18 displays the PF distribution of the first three objective functions of the water resource management problem with five objectives. The result illustrates that MOPSO, MOEOSMA, and MOMPA provide the best PF distribution, but the HV values of MOMVO and MOSMA are larger, indicating that the latter has superior convergence and is closer to the true PF.

For the truss optimization problem, taking 60-bar as an example, Fig. 19 presents the PF obtained by all comparison algorithms. It can be seen that MOEOSMA demonstrates the best convergence performance on the 60-bar truss optimization problem, and the PF obtained is closer to the true PF. Moreover, the PF obtained by many algorithms can only

cover part of the true PF, and only MOEOSMA, MOGWO, and MOMPA can achieve high coverage.

In summary, MOEOSMA shows the strongest competitiveness in all two-objective engineering problems, indicating that the multi-objective algorithm proposed in this research is very efficient in two-objective optimization problems. Among them, the dynamic exploration and exploitation coefficient enhance the search capability of the algorithm, and the elite archiving mechanism based on the crowding distance can promote the convergence and diversity of the PF. For engineering problems with three objectives and above, MOEOSMA's performance is reduced due to too little selection pressure caused by Pareto dominance. However, it still has a strong competitive advantage and outperforms most comparable MOAs.

6 Conclusion and future work

This study developed a novel multi-objective version of the recently proposed EOSMA called MOEOSMA. Here, EOSMA's superior performance in the decision space is the primary motivation for developing MOEOSMA. In order to handle MOPs efficiently, the proposed algorithm introduces three important components. First, dynamic exploration and exploitation coefficients were used to improve the algorithm's search ability in the decision space. Second, a rotation method was used designed to sort the fitness of slime mould individual to evaluate the fitness weight. Then, a Pareto archive with a fixed capacity was used to store the good non-dominated solutions obtained so far to improve the convergence of solutions in the objective space. Finally, a crowding distance assessment method was developed to maintain the archive and update the equilibrium pool to promote the diversity of the solution in the objective space.

The performance of MOEOSMA was verified on CEC2020 functions, and the convergence of the algorithm in the decision space and objective space was evaluated using IGDX and IGDF, respectively. The experimental results show that MOEOSMA outperforms nine well-known MOAs. In addition, eight real-world engineering problems and four truss optimization problems were tested to demonstrate the efficiency and practicality of MOEOSMA. The convergence, diversity and extensiveness of algorithms were evaluated by HV and STE, respectively. In terms of convergence and diversity, MOEOSMA is obviously superior to other comparison algorithms. In terms of extensiveness, MOEOSMA is second only to SPEA2. In addition, MOEOSMA obtained the largest number of non-dominated solutions, which can provide more alternatives to decision-makers. In future research, MOEOSMA can be applied to more practical optimization problems, such as multi-objective feature selection problem (Hu et al. 2022) and wing aeroelastic optimization problem (Wansasueb et al. 2022). In addition, the crowding distance method can be further improved to enhance the performance of MOEOSMA in solving multimodal multi-objective optimization problems.

Appendix 1. Real-world constraint engineering design problems

Speed reducer design problem

Consider $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7] = [b, m, z, l_1, l_2, d_1, d_2]$

Minimize $f_1(\mathbf{x}) = 0.7854x_1x_2^2(14.9334x_3 + 3.3333x_3^2 - 43.0934)$

$-1.508x_1\left(x_6^2 + x_7^2\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right)$

$f_2(\mathbf{x}) = \sqrt{\left(745x_4/(x_2x_3)\right)^2 + 16.9 \times 10^6} \Big/ (0.1x_6^3)$

subject to: $g_1(\mathbf{x}) = 27/(x_1x_3x_2^2) \le 1; g_2(\mathbf{x}) = 397.5/(x_1x_2^2x_3^2) \le 1$

$g_3(\mathbf{x}) = 1.93x_4^3/(x_2x_3x_6^4) \le 1; g_4(\mathbf{x}) = 1.93x_5^3/(x_2x_3x_7^4) \le 1$

$g_5(\mathbf{x}) = x_2x_3/40 \le 1; g_6(\mathbf{x}) = x_1/(12x_2) \le 1$

$g_7(\mathbf{x}) = 5x_2/x_1 \le 1; g_8(\mathbf{x}) = (1.5x_6 + 1.9)/x_4 \le 1$

$g_9(\mathbf{x}) = (1.1x_7 + 1.9)/x_5 \le 1; g_{10}(\mathbf{x}) = f_2(\mathbf{x}) - 1100 \le 1$

$g_{11}(\mathbf{x}) = \sqrt{\left(745x_5/(x_2x_3)\right)^2 + 157.5 \times 10^6} \Big/ (0.1x_7^3) - 850 \le 1$

with $2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28$(integer),

$7.3 \le x_4, x_5 \le 8.3, 2.9 \le x_6 \le 3.9, 5.0 \le x_7 \le 5.5.$

Spring design problem

Consider: $\mathbf{x} = [x_1, x_2, x_3] = [d, D, N]$

Minimize: $f_1(\mathbf{x}) = 0.25\pi^2x_1^2x_2(x_3 + 2)$

$f_2(\mathbf{x}) = 8000c_fx_2/(\pi x_1^3)$

subject to: $g_1(\mathbf{x}) = f_1(\mathbf{x}) - V_{\max} \le 0; g_2(\mathbf{x}) = f_2(\mathbf{x}) - S \le 0$

$g_3(\mathbf{x}) = l_f - l_{\max} \le 0; g_4(\mathbf{x}) = d_{\min} - x_1 \le 0$

$g_5(\mathbf{x}) = x_1 + x_2 - D_{\max} \le 0; g_6(\mathbf{x}) = 3 - C \le 0$

$g_7(\mathbf{x}) = \sigma_p - \sigma_{pm} \le 0; g_8(\mathbf{x}) = 1.25 - 700/K \le 0$

where $V_{\max} = 30; S = 189000; l_{\max} = 14; d_{\min} = 0.2;$

$D_{\max} = 3; \sigma_{pm} = 6; \sigma_p = 300/K;$

$K = Gx_1^4/(8x_2^3x_3); G = 11.5 \times 10^6;$

$c_f = (4C - 1)/(4C - 4) + 0.615/C; C = x_2/x_1;$

$l_f = 1.05x_1(x_3 + 2) + 1000/K.$

with $x_1 \in \{0.009, 0.0095, 0.0104, 0.0118, 0.0128, 0.0132, 0.014,$

$0.015, 0.0162, 0.0173, 0.018, 0.020, 0.023, 0.025,$

$0.028, 0.032, 0.035, 0.041, 0.047, 0.054, 0.063,$

$0.072, 0.080, 0.092, 0.0105, 0.120, 0.135, 0.148,$

$0.162, 0.177, 0.192, 0.207, 0.225, 0.244, 0.263,$

$0.283, 0.307, 0.331, 0.362, 0.394, 0.4375, 0.500\},$

$1 \le x_2 \le 30$(continuous), $1 \le x_3 \le 32$(integer).

Hydrostatic thrust bearing design problem

Consider: $\mathbf{x} = [R, R_0, \mu, Q]$

Minimize: $f_1(\mathbf{x}) = \frac{1}{12}\left(\frac{Q \times P_0}{0.7} + E_f\right)$

$f_2(\mathbf{x}) = \frac{\gamma}{g \cdot P_0} \cdot \frac{Q}{2\pi Rh}$

subject to: $g_1(\mathbf{x}) = W_s - W \le 0; g_2(\mathbf{x}) = P_0 - P_{\max} \le 0$

$g_3(\mathbf{x}) = \Delta T - \Delta T_{\max} \le 0; g_4(\mathbf{x}) = h_{\min} - h \le 0$

$g_5(\mathbf{x}) = R_0 - R \le 0; g_6(\mathbf{x}) = f_2(\mathbf{x}) - 0.001 \le 0$

$g_7(\mathbf{x}) = W \big/ \left(\pi\left(R^2 - R_0^2\right)\right) - 5000 \le 0$

where $W = \frac{\pi P_0}{2} \cdot \frac{R^2 - R_0^2}{\ln\left(R \big/ R_0\right)}; P_0 = \frac{6\mu Q}{\pi h^3} \cdot \ln\left(\frac{R}{R_0}\right);$

$h = \left(\frac{2\pi N}{60}\right)^2 \cdot \frac{2\pi\mu}{E_f} \cdot \frac{R^4 - R_0^4}{4}; E_f = 9336Q \cdot \gamma \cdot C \cdot \Delta T;$

$\Delta T = 2\left(10^P - 560\right); P = \frac{\log_{10}\log_{10}\left(8.122 \times 10^6\mu + 0.8\right) - C_1}{n};$

$\gamma = 0.0307; C = 0.5; n = -3.55; C_1 = 10.04; W_s = 101000;$

$P_{\max} = 1000; \Delta T_{\max} = 50; h_{\min} = 0.001; g = 386.4; N = 750.$

with $1 \le R, R_0, Q \le 16, 1 \times 10^{-6} \le \mu \le 16 \times 10^{-6}$.

Vibrating platform design problem

Consider $\mathbf{x} = [d_1, d_2, d_3, b, L]$

Minimize $f_1(\mathbf{x}) = -\pi \big/ (2L^2) \cdot \sqrt{EI \big/ \mu}$

$f_2(\mathbf{x}) = 2b \cdot L\left(c_1 d_1 - c_2(d_1 - d_2) - c_3(d_2 - d_3)\right)$

subject to: $g_1(\mathbf{x}) = \mu L - 2800 \le 0; g_2(\mathbf{x}) = d_1 - d_2 \le 0$

$g_3(\mathbf{x}) = d_2 - d_1 - 0.15 \le 0; g_4(\mathbf{x}) = d_2 - d_3 \le 0$

$g_5(\mathbf{x}) = d_3 - d_2 - 0.01 \le 0$

where $EI = (2b/3)(E_1 d_1^3 - E_2(d_1^3 - d_2^3) - E_3(d_2^3 - d_3^3));$

$\mu = 2b(\rho_1 d_1 - \rho_2(d_1 - d_2) - \rho_3(d_2 - d_3)).$

with $0.05 \le d_1 \le 0.5, 0.2 \le d_2 \le 0.5, 0.2 \le d_3 \le 0.6,$

$0.35 \le b \le 0.5, 3 \le L \le 6.$

Car side impact design problem

Consider $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$

Minimize $f_1(\mathbf{x}) = 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5$

$+10^{-5}x_6 + 2.73x_7 + 1.98$

$f_2(\mathbf{x}) = 4.72 - 0.19x_2x_3 - 0.50x_4$

$f_3(\mathbf{x}) = 0.50 \cdot (V_{MBP} + V_{FD})$

subject to: $g_1(\mathbf{x}) = 1.16 - 0.0092928x_3 - 0.3717x_2x_4 \le 1$

$g_2(\mathbf{x}) = 0.261 - 0.06486x_1 + 0.0154464x_6 - 0.0159x_1x_2$

$-0.019x_2x_7 + 0.0144x_3x_5 \le 0.32$

$g_3(\mathbf{x}) = 0.214 - 0.0587118x_1 + 0.018x_2^2 + 0.030408x_3$

$+0.00817x_5 + 0.03099x_2x_6 - 0.018x_2x_7$

$-0.00364x_5x_6 \le 0.32$

$g_4(\mathbf{x}) = 0.74 - 0.61x_2 + 0.227x_2^2 - 0.031296x_3$

$-0.031872x_7 \le 0.32$

$g_5(\mathbf{x}) = 28.98 + 3.818x_3 + 1.27296x_6 - 2.68065x_7$

$-4.2x_1x_2 \le 32$

$g_6(\mathbf{x}) = 33.86 - 3.795x_2 + 2.95x_3 - 3.4431x_7 - 5.057x_1x_2$

$+1.45728 \le 32$

$g_7(\mathbf{x}) = 46.36 - 4.4505x_1 - 9.9x_2 \le 32$

$g_8(\mathbf{x}) = f_2(\mathbf{x}) \le 4$

$g_9(\mathbf{x}) = V_{MBP} \le 9.9$

$g_{10}(\mathbf{x}) = V_{FD} \le 15.7$

where $V_{MBP} = 10.58 - 0.67275x_2 - 0.674x_1x_2;$

$V_{FD} = 16.45 - 0.489x_3x_7 - 0.843x_5x_6.$

with $0.5 \le x_1, x_3, x_4 \le 1.5, 0.45 \le x_2 \le 1.35,$

$0.875 \le x_5 \le 2.625, 0.4 \le x_6, x_7 \le 1.2.$

Water resource management problem

Consider $\mathbf{x} = [x_1, x_2, x_3]$

Minimize $f_1(\mathbf{x}) = 106780.37(x_2 + x_3) + 61704.67$

$f_2(\mathbf{x}) = 3000x_1$

$f_3(\mathbf{x}) = 30570 \times 2289x_2 \big/ (0.06 \times 2289)^{0.65}$

$f_4(\mathbf{x}) = 250 \times 2289 \times \exp\left(2.74 - 39.75x_2 + 9.9x_3\right)$

$f_5(\mathbf{x}) = 25\left(1.39 / (x_1x_2) + 4940x_3 - 80\right)$

subject to: $g_1(\mathbf{x}) = 4.94x_3 + 0.00139 / (x_1x_2) \le 1.08$

$g_2(\mathbf{x}) = 1.082x_3 + 0.000306 / (x_1x_2) \le 1.0986$

$g_3(\mathbf{x}) = 49408.24x_3 + 12.307 / (x_1x_2) \le 54051.02$

$g_4(\mathbf{x}) = 8046.33x_3 + 2.098 / (x_1x_2) \le 16696.71$

$g_5(\mathbf{x}) = 7883.39x_3 + 2.138 / (x_1x_2) \le 10705.04$

$g_6(\mathbf{x}) = 1721.26x_3 + 0.417x_1x_2 \le 2136.54$

$g_7(\mathbf{x}) = 631.13x_3 + 0.164 / (x_1x_2) \le 604.48$

with $0.01 \le x_1 \le 0.45, 0.01 \le x_2, x_3 \le 0.1.$

Bulk carriers design problem

Consider $\mathbf{x} = [L, B, D, T, V_k, C_B]$

Minimize $f_1(\mathbf{x}) = (C_c + C_r + C_v)/C_a$

$f_2(\mathbf{x}) = W_{ls}$

$f_3(\mathbf{x}) = -C_a$

subject to: $g_1(\mathbf{x}) = -L/B + 6 \le 0$

$g_2(\mathbf{x}) = L/D - 15 \le 0$

$g_3(\mathbf{x}) = -L/T - 19 \le 0$

$g_4(\mathbf{x}) = T - 0.45D_{wt}^{0.31} \le 0$

$g_5(\mathbf{x}) = T - 0.7D - 0.7 \le 0$

$g_6(\mathbf{x}) = F_n - 0.32 \le 0$

$g_7(\mathbf{x}) = -0.53T - ((0.085C_B - 0.002)B^2)/(T \cdot C_B)$

$+(1 + 0.52D) + 0.07B \le 0$

$g_8(\mathbf{x}) = -D_{wt} + 3000 \le 0;$

$g_9(\mathbf{x}) = D_{wt} - 500000 \le 0$

where $C_c = 2.6(2000W_s^{0.85} + 3500W_o + 2400P^{0.8});$

$C_r = 40000D_{wt}^{0.3}; C_v = (105D_cS_d + 6.3D_{wt}^{0.8})R_{tpa};$

$C_a = D_{cwt}R_{tpa}; R_{tpa} = 350/(S_d + 2(D_{cwt}/8000 + 0.5));$

$D_{cwt} = D_{wt} - D_c(S_d + 5) - 2D_{wt}^{0.5}; S_d = 5000V_k/24;$

$D_c = 0.19 \times 24P/1000 + 0.2; D_{wt} = 1.025L \cdot B \cdot T \cdot C_B - W_{ls};$

$W_{ls} = W_s + W_o + W_m; W_s = 0.034L^{1.7}B^{0.7}D^{0.4}C_B^{0.5};$

$W_o = L^{0.8}B^{0.6}D^{0.3}C_B^{0.1}; W_m = 0.17P^{0.9};$

$P = (1.025L \cdot B \cdot T \cdot C_B)^{\frac{2}{3}}V_k^3/(a + b \cdot F_n);$

$F_n = 0.5144V_k/(9.8065L)^{0.5};$

$a = 4456.51 - 8105.61C_B + 4977.06C_B^2;$

$b = -6960.32 + 12817C_B - 10847.2C_B^2.$

with $150.0 \le L \le 274.32, 20.0 \le B \le 32.31,$

$13.0 \le D \le 25.0, 10.0 \le T \le 11.71,$

$14.0 \le V_k \le 18.0, 0.63 \le C_B \le 0.75.$

Multi-product batch plant problem

Consider $\mathbf{x} = [N_1, N_2, N_3, V_1, V_2, V_3, T_{L1}, T_{L2}, B_1, B_2]$

Minimize $f_1(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j N_j V_j^{\beta_j}$

$f_2(\mathbf{x}) = 65 \sum_{i=1}^{N} \frac{Q_i}{B_i} + 0.08Q_1 + 0.1Q_2$

$f_3(\mathbf{x}) = \sum_{i=1}^{N} \frac{Q_i T_{Li}}{B_i}$

subject to: $g_1(\mathbf{x}) = f_3(\vec{x}) - H \le 0$

$g_2(\mathbf{x}) = \sum_{i=1}^{N} S_{ij}B_i - V_j \le 0, j = 1, ..., M$

$g_3(\mathbf{x}) = t_{ij} - N_j T_{Li} \le 0, i = 1, ..., N; j = 1, ..., M$

where $N = 2; M = 3; \alpha_j = 250; \beta_j = 0.6;$

$H = 6000; Q_1 = 40000; Q_2 = 20000;$

$S_{11} = 2; S_{12} = 3; S_{13} = 4;$

$S_{21} = 4; S_{22} = 6; S_{23} = 3;$

$t_{11} = 8; t_{12} = 20; t_{13} = 8;$

$t_{21} = 16; t_{22} = 4; t_{23} = 4.$

with $1 \le N_1, N_2, N_3 \le 3$(integer), $250 \le V_1, V_2, V_3 \le 2500,$

$6 \le T_{L1} \le 20, 4 \le T_{L2} \le 16, 40 \le B_1 \le 700, 10 \le B_2 \le 450.$

Appendix 2. Pareto fronts obtained by all comparison algorithms

See Figs. 20, 21, 22, 23, 24, 25, 26, 27

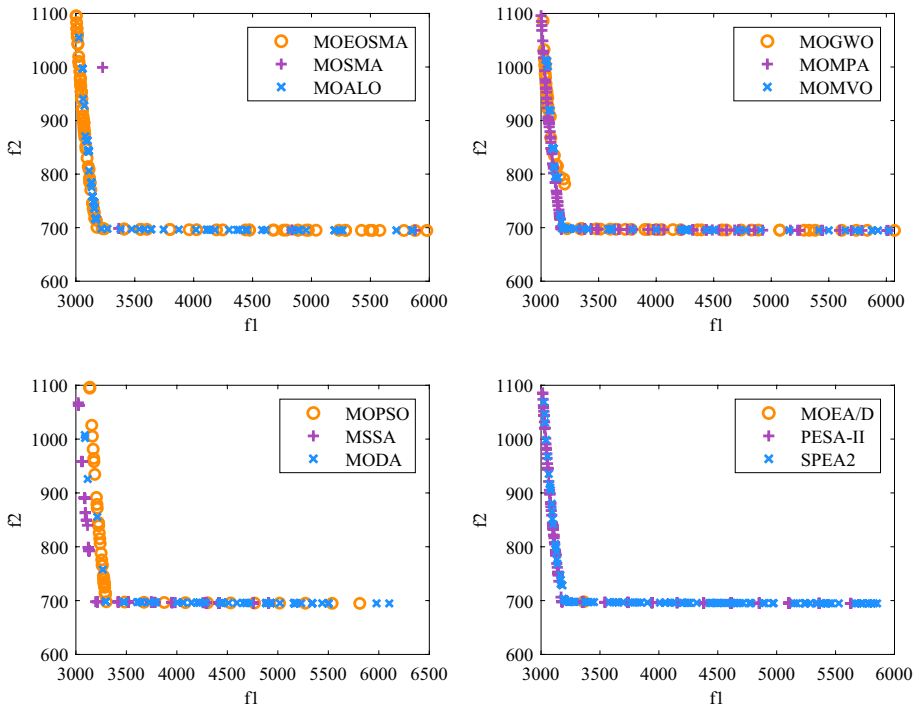


Fig. 20 Speed reducer design problem

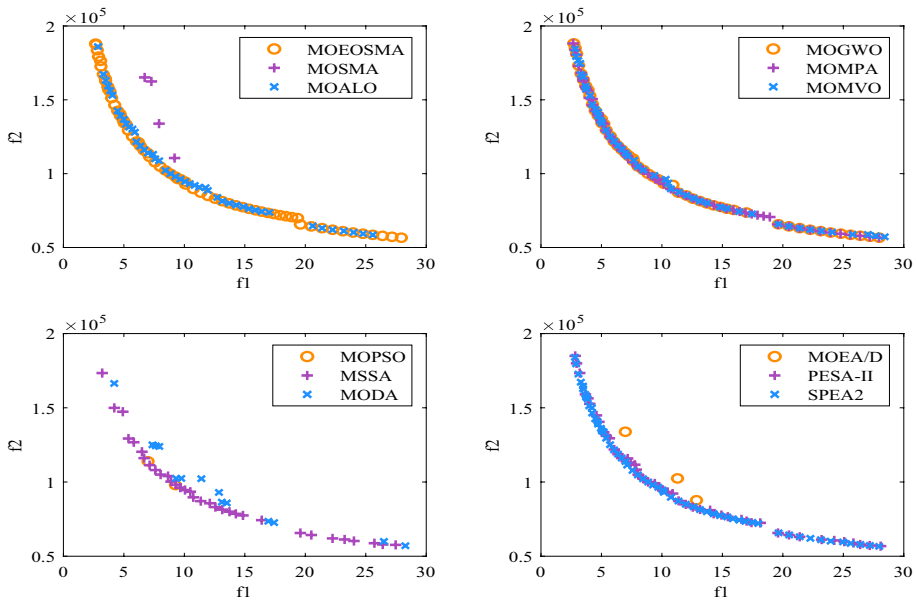


Fig. 21 Spring design problem

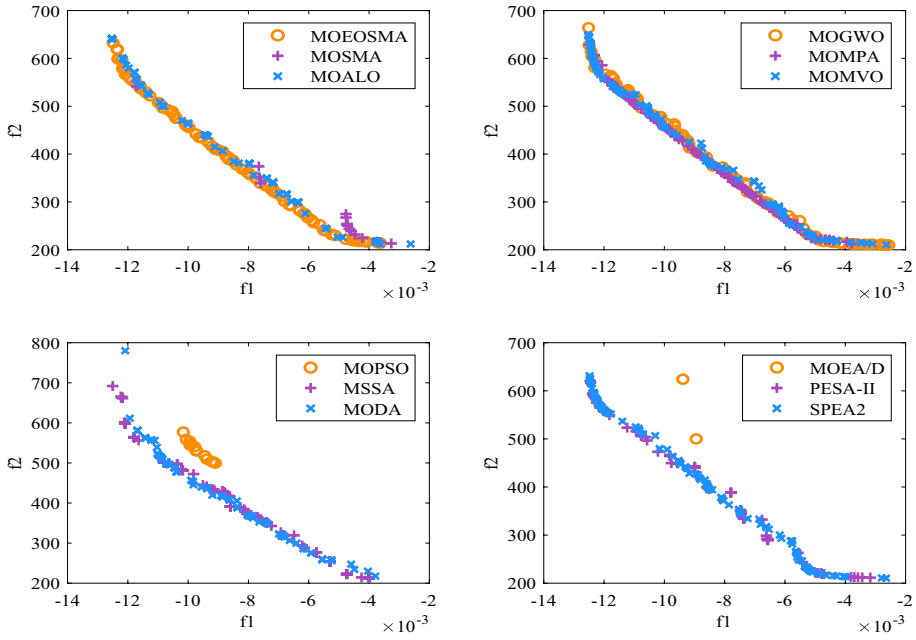


Fig. 22 Vibrating platform design problem

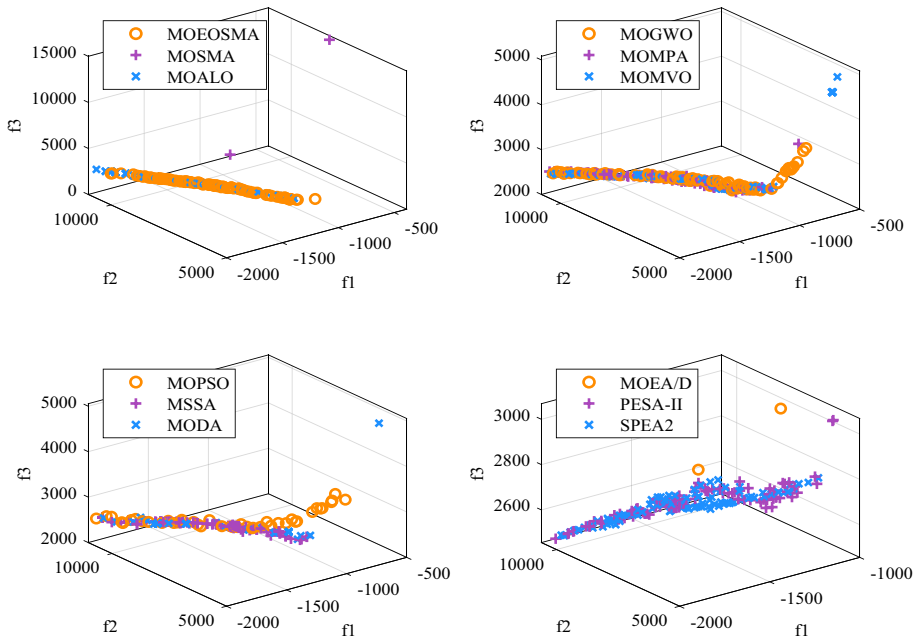


Fig. 23 Bulk carriers design problem

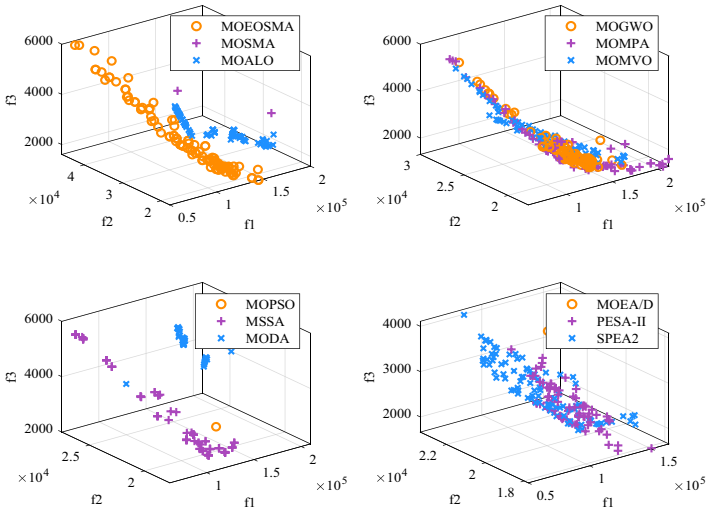


Fig. 24 Multi-product batch plant design problem

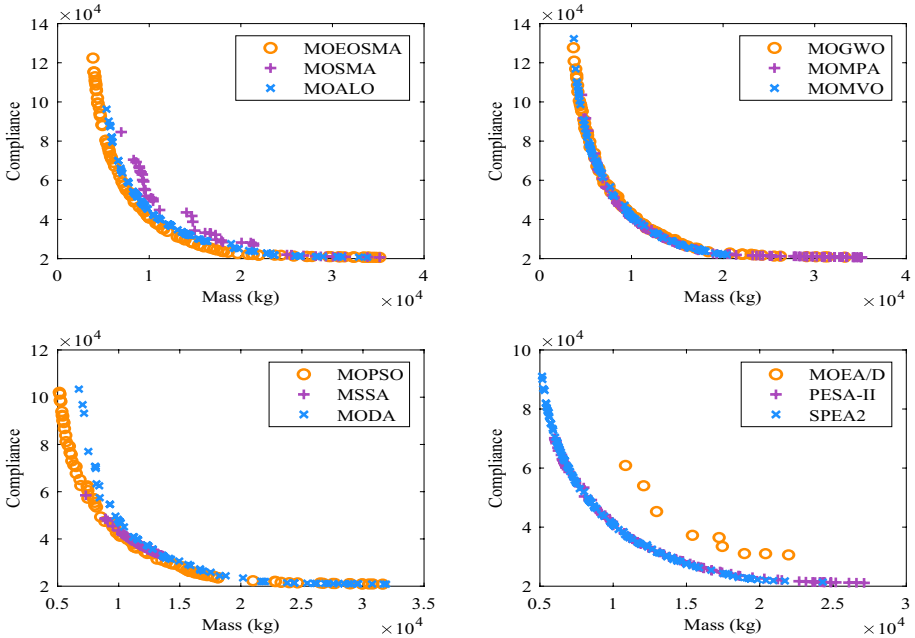


Fig. 25 72-bar 3D truss optimization problem

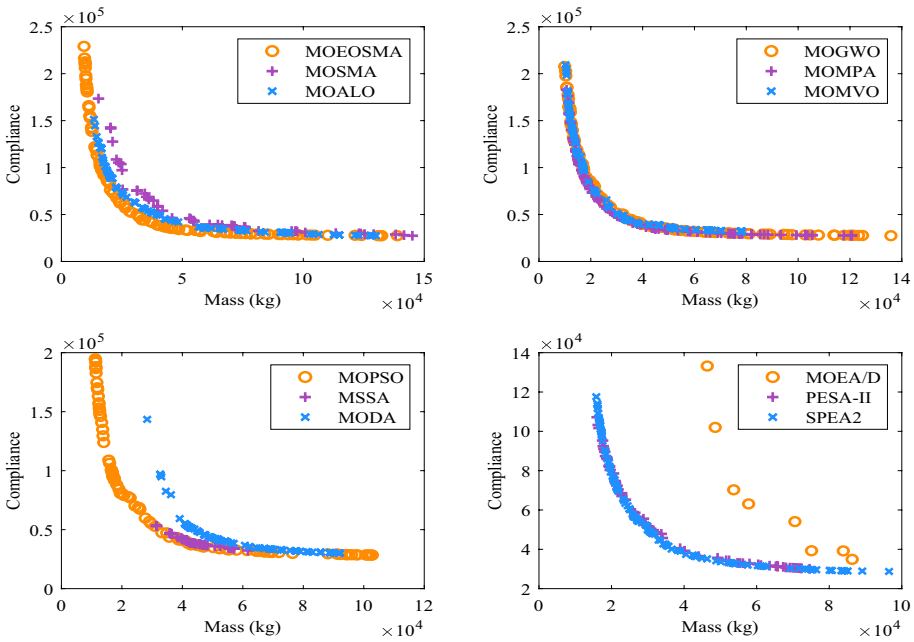


Fig. 26 200-bar 2D truss optimization problem

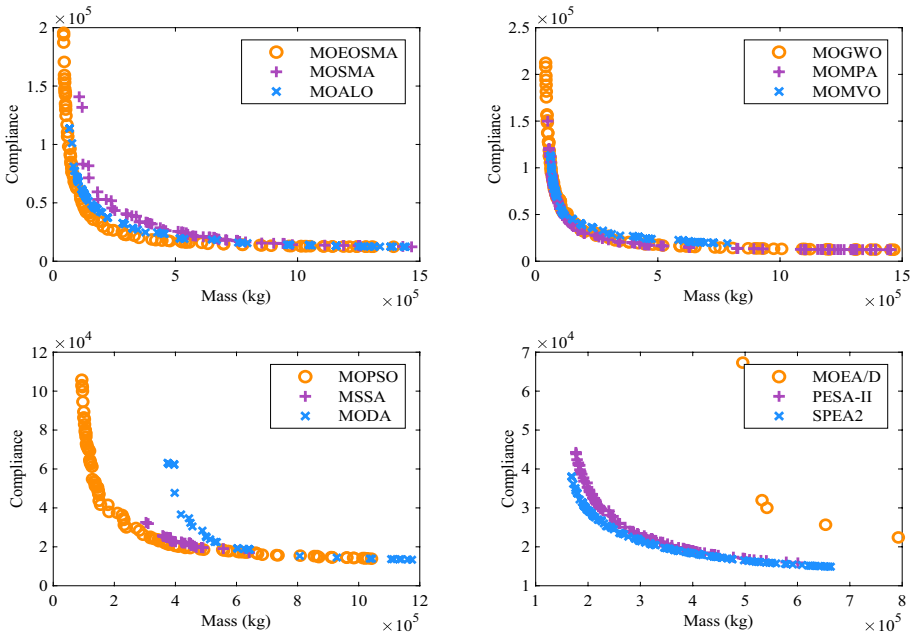


Fig. 27 942-bar 3D truss optimization problem

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grants Nos. 62066005, U21A20464, 62102183, and Project of the Jiangsu Province Natural Science Foundation under Grant under Grant No. BK20180462. Project of the Guangxi Science and Technology under Grant No. 2019KY0185, and Innovation Project of Guangxi Minzu University Graduate Education under Grant gxun-chxs2021058.

Author contributions QL: validation, writing—review & editing, supervision. SY: conceptualization, methodology, writing—original draft. GZ: algorithm design & analysis. WM: algorithm analysis. YZ: review & editing. YZ: supervision, writing—review & editing.

Data availability All data, models, and code generated or used during the study appear in the submitted article.

Declarations

Conflict of interest No potential conflict of interest has been stated by the authors.

Replication of results GitHub: https://github.com/Shihong-Yin/MOEOSMA

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Ali MA, Shimoda M (2022) Toward multiphysics multiscale concurrent topology optimization for lightweight structures with high heat conductivity and high stiffness using MATLAB. Struct Multidisc Optim 65:207. https://doi.org/10.1007/s00158-022-03291-0

Ayala HVH, Klein CE, Mariani VC, Coelho LDS (2017) Multiobjective symbiotic search algorithm approaches for electromagnetic optimization. IEEE Trans Magn 53:1–4. https://doi.org/10.1109/TMAG.2017.2665350

Becker M (2015) On the efficiency of nature-inspired algorithms for generation of fault-tolerant graphs. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. IEEE, Kowloon Tong, Hong Kong, pp 1657–1663

Benaissa B, Hocine NA, Khatir S, Riahi MK, Mirjalili S (2021) YUKI algorithm and POD-RBF for elastostatic and dynamic crack identification. J Comput Sci 55:101451. https://doi.org/10.1016/j.jocs.2021.101451

Branke J, Deb K, Dierolf H, Osswald M (2004) Finding knees in multi-objective optimization. Parallel problem solving from nature—PPSN VIII. Springer, Berlin, pp 722–731

Champasak P, Panagant N, Pholdee N, Bureerat S, Yildiz AR (2020) Self-adaptive many-objective meta-heuristic based on decomposition for many-objective conceptual design of a fixed wing unmanned aerial vehicle. Aerosp Sci Technol 100:105783. https://doi.org/10.1016/j.ast.2020.105783

Chen Z, Liu W (2020) An efficient parameter adaptive support vector regression using K-means clustering and chaotic slime mould algorithm. IEEE Access 8:156851–156862. https://doi.org/10.1109/ACCESS.2020.3018866

Chen J, Luo Q, Zhou Y, Huang H (2022) Firefighting multi strategy marine predators algorithm for the early-stage forest fire rescue problem. Appl Intell. https://doi.org/10.1007/s10489-022-04265-x

Chou J-S, Truong D-N (2020) Multiobjective optimization inspired by behavior of jellyfish for solving structural design problems. Chaos Solitons Fractals 135:109738. https://doi.org/10.1016/j.chaos.2020.109738

Coello CAC (2009) Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. Front Comput Sci China 3:18–30. https://doi.org/10.1007/s11704-009-0005-7

Coello CAC, Lechuga MS (2002) MOPSO: A proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). IEEE, Honolulu, pp 1051–1056

Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8:256–279. https://doi.org/10.1109/TEVC.2004.826067

Corne DW, Jerram NR, Knowles JD, Oates MJ (2001) PESA-II: Region-based selection in evolutionary multiobjective optimization. In: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, GECCO'01. Morgan Kaufmann Publishers, San Francisco, pp 283–290

Cui X, Luo Q, Zhou Y, Deng W, Yin S (2022) Quantum-inspired moth-flame optimizer with enhanced local search strategy for cluster analysis. Front Bioeng Biotechnol 10:908356. https://doi.org/10.3389/fbioe.2022.908356

Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints. IEEE Trans Evol Comput 18:577–601. https://doi.org/10.1109/TEVC.2013.2281535

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

Dhiman G, Kumar V (2018) Multi-objective spotted hyena optimizer: a multi-objective optimization algorithm for engineering problems. Knowl-Based Syst 150:175–197. https://doi.org/10.1016/j.knosys.2018.03.011

Dhiman G, Singh KK, Soni M, Nagar A, Dehghani M, Slowik A, Kaur A, Sharma A, Houssein EH, Cengiz K (2021) MOSOA: a new multi-objective seagull optimization algorithm. Expert Syst Appl 167:114150. https://doi.org/10.1016/j.eswa.2020.114150

Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2020) Equilibrium optimizer: a novel optimization algorithm. Knowl-Based Syst 191:105190. https://doi.org/10.1016/j.knosys.2019.105190

Gong D, Xu B, Zhang Y, Guo Y, Yang S (2020) A similarity-based cooperative co-evolutionary algorithm for dynamic interval multiobjective optimization problems. IEEE Trans Evol Comput 24:142–156. https://doi.org/10.1109/TEVC.2019.2912204

Hancer E, Xue B, Zhang M, Karaboga D, Akay B (2015) A multi-objective artificial bee colony approach to feature selection using fuzzy mutual information. In: 2015 IEEE Congress on Evolutionary Computation (CEC). IEEE, Sendai, pp 2420–2427

Hassan MH, Kamel S, Abualigah L, Eid A (2021) Development and application of slime mould algorithm for optimal economic emission dispatch. Expert Syst Appl 182:115205. https://doi.org/10.1016/j.eswa.2021.115205

Houssein EH, Mahdy MA, Shebl D, Manzoor A, Sarkar R, Mohamed WM (2022) An efficient slime mould algorithm for solving multi-objective optimization problems. Expert Syst Appl 187:115870. https://doi.org/10.1016/j.eswa.2021.115870

Hu Y, Wang J, Liang J, Wang Y, Ashraf U, Yue C, Yu K (2022) A two-archive model based evolutionary algorithm for multimodal multi-objective optimization problems. Appl Soft Comput 119:108606. https://doi.org/10.1016/j.asoc.2022.108606

Jain H, Deb K (2014) An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: handling constraints and extending to an adaptive approach. IEEE Trans Evol Comput 18:602–622. https://doi.org/10.1109/TEVC.2013.2281534

Jin Y, Olhofer M, Sendhoff B (2001) Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? In: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, San Francisco, pp 1042–1049

Kollat JB, Reed P (2007) A framework for visually interactive decision-making and design using evolutionary multi-objective optimization (VIDEO). Environ Model Softw 22:1691–1704. https://doi.org/10.1016/j.envsoft.2007.02.001

Kumar A, Wu G, Ali MZ, Luo Q, Mallipeddi R, Suganthan PN, Das S (2021a) A benchmark-suite of real-world constrained multi-objective optimization problems and some baseline results. Swarm Evol Comput 67:100961. https://doi.org/10.1016/j.swevo.2021.100961

Kumar S, Tejani GG, Pholdee N, Bureerat S (2021b) Multi-objective modified heat transfer search for truss optimization. Eng Comput 37:3439–3454. https://doi.org/10.1007/s00366-020-01010-1

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

Li M, Zheng J (2009) Spread assessment for evolutionary multi-objective optimization. In: Ehrgott M, Fonseca CM, Gandibleux X, Hao

J-K, Sevaux M (eds) Evolutionary multi-criterion optimization. Springer, Berlin, pp 216–230
Li K, Torres CE, Thomas K, Rossi LF, Shen C-C (2011) Slime mold inspired routing protocols for wireless sensor networks. Swarm Intell 5:183–223. https://doi.org/10.1007/s11721-011-0063-y
Li S, Chen H, Wang M, Heidari AA, Mirjalili S (2020) Slime mould algorithm: a new method for stochastic optimization. Future Gener Comput Syst 111:300–323. https://doi.org/10.1016/j.future.2020.03.055
Liang JJ, Suganthan PN, Qu BY, Gong DW, Yue CT (2020) Problem definitions and evaluation criteria for the CEC 2020 special session on multimodal multiobjective optimization.
Liu Y, Heidari AA, Ye X, Liang G, Chen H, He C (2021) Boosting slime mould algorithm for parameter identification of photovoltaic models. Energy 234:121164. https://doi.org/10.1016/j.energy.2021.121164
Messac A (1996) Physical programming: effective optimization for computational design. AIAA J 34:149–158. https://doi.org/10.2514/3.13035
Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27:1053–1073. https://doi.org/10.1007/s00521-015-1920-1
Mirjalili S, Saremi S, Mirjalili SM, dos Coelho L (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119. https://doi.org/10.1016/j.eswa.2015.10.039
Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017a) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
Mirjalili S, Jangir P, Mirjalili SZ, Saremi S, Trivedi IN (2017b) Optimization of problems with multiple objectives using the multi-verse optimization algorithm. Knowl-Based Syst 134:50–71. https://doi.org/10.1016/j.knosys.2017.07.018
Mirjalili S, Jangir P, Saremi S (2017c) Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Appl Intell 46:79–95. https://doi.org/10.1007/s10489-016-0825-8
Mirjalili SZ, Mirjalili S, Saremi S, Faris H, Aljarah I (2018) Grasshopper optimization algorithm for multi-objective optimization problems. Appl Intell 48:805–820. https://doi.org/10.1007/s10489-017-1019-8
Musselman K, Talavage J (1980) A tradeoff cut approach to multiple objective optimization. Oper Res 28:1424–1435. https://doi.org/10.1287/opre.28.6.1424
Naik MK, Panda R, Abraham A (2022) Normalized square difference based multilevel thresholding technique for multispectral images using leader slime mould algorithm. J King Saud Univ—Comput Inf Sci 34:4524–4536. https://doi.org/10.1016/j.jksuci.2020.10.030
Narayanan S, Azarm S (1999) On improving multiobjective genetic algorithms for design optimization. Struct Optim 18:146–155
Panagant N, Pholdee N, Bureerat S, Yildiz AR, Mirjalili S (2021) A comparative study of recent multi-objective metaheuristics for solving constrained truss optimisation problems. Arch Comput Methods Eng 28:4031–4047. https://doi.org/10.1007/s11831-021-09531-8
Parsons MG, Scott RL (2004) Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. J Ship Res 48:61–76. https://doi.org/10.5957/jsr.2004.48.1.61
Pholdee N, Bureerat S (2013) Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. Inf Sci 223:136–152. https://doi.org/10.1016/j.ins.2012.10.008
Premkumar M, Jangir P, Sowmya R (2021a) MOGBO: a new multiobjective gradient-based optimizer for real-world structural optimization problems. Knowl-Based Syst 218:106856. https://doi.org/10.1016/j.knosys.2021.106856
Premkumar M, Jangir P, Sowmya R, Alhelou HH, Heidari AA, Chen H (2021b) MOSMA: Multi-objective slime mould algorithm based on elitist non-dominated sorting. IEEE Access 9:3229–3248. https://doi.org/10.1109/ACCESS.2020.3047936
Qi Y, Ma X, Liu F, Jiao L, Sun J, Wu J (2014) MOEA/D with adaptive weight adjustment. Evol Comput 22:231–264. https://doi.org/10.1162/EVCO_a_00109
Qian T, Zhang Z, Gao C, Wu Y, Liu Y (2013) An ant colony system based on the Physarum network. In: Tan Y, Shi Y, Mo H (eds) Advances in swarm intelligence. Springer, Berlin, pp 297–305
Rao RV, Savsani VJ (2012) Mechanical design optimization using advanced optimization techniques. Springer Science & Business Media, London
Ray T, Tai K, Seow KC (2001) Multiobjective design optimization by an evolutionary algorithm. Eng Optim 33:399–424. https://doi.org/10.1080/03052150108940926
Sadollah A, Eskandar H, Kim JH (2015) Water cycle algorithm for solving constrained multi-objective optimization problems. Appl Soft Comput 27:279–298. https://doi.org/10.1016/j.asoc.2014.10.042
Savsani P, Savsani V (2016) Passing vehicle search (PVS): a novel metaheuristic algorithm. Appl Math Model 40:3951–3978. https://doi.org/10.1016/j.apm.2015.10.040
Tawhid MA, Savsani V (2019) Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems. Neural Comput Appl 31:915–929. https://doi.org/10.1007/s00521-017-3049-x
Tejani GG, Pholdee N, Bureerat S, Prayogo D, Gandomi AH (2019) Structural optimization using multi-objective modified adaptive symbiotic organisms search. Expert Syst Appl 125:425–441. https://doi.org/10.1016/j.eswa.2019.01.068
Tero A, Takagi S, Saigusa T, Ito K, Bebber DP, Fricker MD, Yumiki K, Kobayashi R, Nakagaki T (2010) Rules for biologically inspired adaptive network design. Science 327:439–442. https://doi.org/10.1126/science.1177894
Wansasueb K, Pholdee N, Panagant N, Bureerat S (2022) Multiobjective meta-heuristic with iterative parameter distribution estimation for aeroelastic design of an aircraft wing. Eng Comput 38:695–713. https://doi.org/10.1007/s00366-020-01077-w
Wazery YM, Saber E, Houssein EH, Ali AA, Amer E (2021) An efficient slime mould algorithm combined with K-nearest neighbor for medical classification tasks. IEEE Access 9:113666–113682. https://doi.org/10.1109/ACCESS.2021.3105485
Wei Y, Zhou Y, Luo Q, Deng W (2021) Optimal reactive power dispatch using an improved slime mould algorithm. Energy Rep 7:8742–8759. https://doi.org/10.1016/j.egyr.2021.11.138
Wei Y, Othman Z, Daud KM, Yin S, Luo Q, Zhou Y (2022) Equilibrium optimizer and slime mould algorithm with variable neighborhood search for job shop scheduling problem. Mathematics 10:4063. https://doi.org/10.3390/math10214063
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82. https://doi.org/10.1109/4235.585893
Yin S, Luo Q, Du Y, Zhou Y (2022a) DTSMA: dominant swarm with adaptive t-distribution mutation-based slime mould algorithm. Math Biosci Eng 19:2240–2285. https://doi.org/10.3934/mbe.2022105
Yin S, Luo Q, Zhou Y (2022b) EOSMA: An equilibrium optimizer slime mould algorithm for engineering design problems. Arab J Sci Eng 47:10115–10146. https://doi.org/10.1007/s13369-021-06513-7

Yue C, Qu B, Yu K, Liang J, Li X (2019) A novel scalable test problem suite for multimodal multiobjective optimization. Swarm Evol Comput 48:62–71. https://doi.org/10.1016/j.swevo.2019.03.011

Zeng N, Song D, Li H, You Y, Liu Y, Alsaadi FE (2021) A competitive mechanism integrated multi-objective whale optimization algorithm with differential evolution. Neurocomputing 432:170–182. https://doi.org/10.1016/j.neucom.2020.12.065

Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11:712–731. https://doi.org/10.1109/TEVC.2007.892759

Zhang Q, Zhou A, Jin Y (2008) RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm. IEEE Trans Evol Comput 12:41–63. https://doi.org/10.1109/TEVC.2007.894202

Zhao W, Zhang Z, Mirjalili S, Wang L, Khodadadi N, Mirjalili SM (2022) An effective multi-objective artificial hummingbird algorithm with dynamic elimination-based crowding distance for solving engineering design problems. Comput Methods Appl Mech Eng. https://doi.org/10.1016/j.cma.2022.115223

Zhong K, Zhou G, Deng W, Zhou Y, Luo Q (2021) MOMPA: multi-objective marine predator algorithm. Comput Methods Appl Mech Eng. https://doi.org/10.1016/j.cma.2021.114029

Zhou A, Zhang Q, Jin Y (2009) Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. IEEE Trans Evol Comput 13:1167–1189. https://doi.org/10.1109/TEVC.2009.2021467

Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. Parallel problem solving from nature—PPSN VIII. Springer, Berlin, pp 832–842

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength pareto evolutionary algorithm. TIK-Rep 103:1–21. https://doi.org/10.3929/ETHZ-A-004284029

Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7:117–132. https://doi.org/10.1109/TEVC.2003.810758