Soft Computing (2020) 24:18379–18398
https://doi.org/10.1007/s00500-020-05046-9

METHODOLOGIES AND APPLICATION

MoSSE: a novel hybrid multi-objective meta-heuristic algorithm for engineering design problems

Gaurav Dhiman[1] · Meenakshi Garg[1]

Published online: 13 June 2020

Abstract

This paper introduces a novel hybrid optimization algorithm called *MoSSE* by combining the features of Multi-objective Spotted Hyena Optimizer (*MOSHO*), Salp Swarm Algorithm (*SSA*), and Emperor Penguin Optimizer (EPO). *MoSSE* uses *MOSHO's* searching capabilities to effectively discover the search space, *SSA's* leading and selection process to achieve the fittest global solution with quicker convergence technique, and *EPO's* effective mover technique for better adjustment of the next solution. The algorithm is tested on ten *IEEE CEC-9* standard test functions and compared with seven well-known multi-objective optimization algorithms according to their performance. The experimental results show that *MoSSE* provides highly competitive outcomes in terms of convergence speed, searchability, and accuracy. Statistical testing is also performed on *IEEE CEC-9* test functions. Four performance metrics (i.e., Hypervolume, Δ_p, Spread, and Epsilon) are used to validate the searching capability of the proposed algorithm. *MoSSE* is further applied to welded beam, multi-disk clutch brake, pressure vessel, 25-bar truss design problems to test its effectiveness. The findings show the utility of the proposed algorithm to resolve the real-life complex multi-objective optimization problems.

Keywords Spotted Hyena optimizer · Salp Swarm Algorithm · Emperor Penguin Optimizer · Multi-objective optimization · Engineering design problems

1 Introduction

Meta-heuristic optimization approaches take enormous interest from researchers over the past few decades in explaining the search and optimization problems. Such approaches aim to find the solution in computationally traceable ways and are comparatively cheaper and more rapidly than comprehensive searching (Bandarua and Kalyanmoy 2016; Dhiman and Amandeep 2019). Nowadays, researchers are trying to develop the meta-heuristic-based approaches for solving complex problems (Kaur and Gaurav 2019; Garg and Dhiman 2020; Verma et al. 2018; Singh and Gaurav 2017; Garg and Dhiman 2020; Chandrawat et al. 2017; Dehghani et al. 2020). Meta-heuristic optimization techniques can be broadly categorized into two categories such as *single-objective* and *multi-objective* (Dhiman and Kaur 2020; Dhiman et al. 2020a, b). *Single-objective* approaches seek to provide the unique global solution after optimizing the key objective function (Chiandussi et al. 2012; Satnam Kaur et al. 2020). However, most real optimization problems require *multi-objective* that need to be tackled at the same time. Since these objectives are usually contradictory in nature, finding the optimal solution for all of the objectives is difficult (Dehghani et al. 2019; Dhiman 2019a, b). *Multi-objective* approaches are also known in these situations, where several objectives need to be reached before some rational conclusion can be drawn (Branke et al. 2004). *Multi-objective* optimization methods also known as vector optimization approach and involves several competing objects and aim to find the fittest possible solutions (Chiandussi et al. 2012; Coello Coello et al. 2007). The key challenge in *multi-objective* optimization is to model the preferences of decision-makers in ordering or assessing the relative importance of competing objectives. This problem has three main approaches, specifically *priori*, *posteriori*, and *interactive* (Marler and Arora 2004). *Priori* approaches employ the scalar function

Communicated by V. Loia.

✉ Gaurav Dhiman
gdhiman0001@gmail.com

Meenakshi Garg
meenagarg82@gmail.com

[1] Department of Computer Science, Government Bikram College of Commerce, Patiala, Punjab 147001, India

to convert the single-objective problem into multi-objective problem (Konak et al. 2006). *Posteriori* approaches may not require the user preference information. These methods provide a range of optimal solutions which are mathematically equivalent to *Pareto* (Dhiman and Kumar 2019). *Interactive* methods are called as the human-in-the-loop technique, constantly gather decision-makers' favorites and contain them in the optimization process to find the *Pareto* optimal solutions (Marler and Arora 2004; Singh et al. 2019; Dhiman 2019).

In 1984, David Schaffer developed Vector Evaluated Genetic Algorithm (*VEGA*) (Coello Coello et al. 2007). It is the first multi-objective optimization algorithm. Apart from this, various multi-objective algorithms were also developed. These are Non-dominated Sorting Genetic Algorithm 2 (*NSGA-II*) (Deb et al. 2002), Multi-objective Evolutionary Algorithm based on Decomposition (*MOEA/D*) (Zhang and Li 2007), Pareto Envelope-based Selection Algorithm 2 (*PESA-II*) (Corne et al. 2001), Multi-objective Particle Swarm Optimization (*MOPSO*) (Coello Coello 2002), Strength Pareto Evolutionary algorithm 2 (*SPEA2*) (Zitzler et al. 2001), Multi-objective Ant Colony Optimization (*MOACO*) (Angus and Woodward 2009), Multi-objective Brain Storm Optimization Algorithm based on Decomposition (*MBSO/D*) (Dai and Lei 2019), Modified Multi-objective Self-Adaptive Differential Evolution Algorithm (*MMOSADE*) (Yang et al. 2019), and Multi-objective Variant of the Vibrating Particles System (*MOVPS*) (Kaveh and Ghazaan 2019), and Multi-objective Salp Swarm Algorithm (*MSSA*) (Mirjalili et al. 2017). These algorithms can resolve a variety of multi-objective problems related to several domains, such as software engineering (Mansoor et al. 2017), quantum computing (Singh et al. 2018; Kaur et al. 2018; Dhiman and Kumar 2018), fuzzy systems (Singh and Dhiman 2018a, b; Singh et al. 2018), computer engineering (Chen and Hammami 2015; Jianjie et al. 2017; Hou et al. 2018; Yu et al. 2015), power systems (Dhiman et al. 2018, 2019), industrial problems (Dhiman and Kaur 2019; Dhiman and Kumar 2019; Dhiman and Kaur 2018), civil engineering (Luh and Chueh 2004), mechanical engineering (Mirjalili et al. 2017; Dhiman and Kumar 2017), and so on.

According to No-Free Lunch Theorem (*NFL*) (Wolpert and Macready 1997), these algorithms are not able to solve all optimization problems. NFL (Wolpert and Macready 1997) has scientifically proved the aforementioned encourages and claim researchers to build new optimization algorithms. There is also space for designing new algorithms with the goal of solving complex problems more efficiently, which are difficult to resolve with present multi-objective approaches. Based on this inspiration, we propose a multi-objective optimization algorithm. This algorithm employs the features of Multi-objective Spotted Hyena Optimizer (*MOSHO*) (Dhiman and Kumar 2018; Dhiman and Kaur 2017; Dhiman and Kumar 2019), Salp Swarm Algorithm (SSA) (Mirjalili et al. 2017), and Emperor Penguin Optimizer (*EPO*) (Dhiman and Kumar 2018). It is termed as Multi-objective Spotted hyena, Salp swarm, and Emperor penguin optimizer (*MoSSE*). *MOSHO*, *SSA*, and *EPO* have proved the effectiveness in resolving various engineering problems. The strengths and drawbacks of these three algorithms are: *MOSHO* has stronger optimization capabilities and removes the problem of missing selection pressure. But the algorithm has slow convergence speed in its phase of *exploitation*. On the other hand, *SSA* and *EPO* prevent a solution from trapping in local optima. These algorithms provide the optimal solutions with high convergence speed. The main contributions to this work can be summarized as follows:

- The *MoSSE* algorithm integrates the archive to accumulate all the non-dominated optimal solutions.
- A selection system for replacing and updating the results in the archive with respect to other solution has been implemented.
- *MoSSE* implements an adaptive grid approach to remove a solution from the most crowded section and increase the variety of non-dominated solutions.

The suggested multi-objective algorithm is evaluated on ten *IEEE CEC-9* benchmark test functions including (Zhang et al. 2008), and four real-life engineering design problems. Also, the efficiency of the suggested algorithm is validated by comparing the results with seven well- and widely used multi-objective optimization algorithms, i.e., *MOPSO* (Coello Coello 2002), *NSGA-II* (Deb et al. 2002), *MOEA/D* (Zhang and Li 2007), *PESA-II* (Corne et al. 2001), *MSSA* (Mirjalili et al. 2017), *MOACO* (Angus and Woodward 2009), and *MOSHO* (Dhiman and Kumar 2018) by using four performance measures.

The remainder section of this paper is presented as follows: Sect. 2 presents a literature review of multi-objective optimization techniques. Section 3 discusses the concepts of *MOSHO*, *SSA*, and *EPO*, and the proposed *MoSSE* algorithms. The experimental observations and reviews are presented in more detail in Sect. 4. Section 5 assesses the proposed algorithm for four engineering design applications by relating its efficiency with seven well-known multi-objective optimization algorithms. Section 6 eventually completes this analysis and outlines the potential direction for future study.

2 Literature review

Multi-objective optimization provides the correct estimate for the non-dominated *Pareto* solutions with maximum variation in a single run (Zhou et al. 2011). Because of its numerous advantages such as local optimum avoidance and gradient-free techniques (Mirjalili et al. 2016), academia and the tech industry have attracted considerable interest in solving many real-life problems. Extensive research has also been conducted over the past three decades to develop and test the multi-objective optimization approaches. Some well-established methods of stochastic multi-objective optimization include *PESA-II* (Corne et al. 2001), *NSGA-II* (Deb et al. 2002), *MOEA/D* (Zhang and Li 2007), *MOPSO* (Coello Coello 2002), *SPEA2* (Zitzler et al. 2001), and *MOACO* (Angus and Woodward 2009).

SPEA2 (Zitzler et al. 2001) is proposed and discussed in 2001. It is the modernized version of *SPEA* (Zitzler and Thiele 1998) which integrates finest-fitness information, fixed-size external archive to store non-dominated solutions, and upgraded clustering method to remove archive members. *PESA-II* (Corne et al. 2000) is the newest version of *PESA* (Corne et al. 2001) algorithm with directed region-based selection method which selects optimal solutions for a hyper-box rather than individual *Pareto* solutions. The sparse populated hyper-boxes take higher chances of being picked during selection process than crowded hyper-boxes. Instead, individual solutions are randomly selected from the selected hyper-box to achieve crossover and mutation. So, it minimizes the cost of measuring which corresponds to the *Pareto* rankings.

Another widely used algorithm is *NSGA-II* (Deb et al. 2002), which is used to resolve the deficiency of elitism, high computational difficulty, and parameter-issue of *NSGA* (Srinivas and Deb 1994). This algorithm uses a fast, non-dominated sorting method, elitist approach, and niching of operators. First, *NSGA-II* creates the random population of competing individuals and then uses non-sorting mechanisms to rank and sort each individual. Growing individual will then be assigned a fitness value that is equal to their non-domination level.

MOPSO (Coello Coello 2002) uses an external pool to store and retrieve the *Pareto* solutions that are more suitable, based on the basic principles of *PSO* (Eberhart and Kennedy 1995). This algorithm determines the best global particle for each member of the population also known as the best local guide from the set of optimal solutions obtained by *Pareto*. This procedure improves the variety of solutions and the speed of convergence, especially when dealing with multi-objective problems.

MOEA/D (Zhang and Li 2007) divides the multi-objective problem into a set of single-objective sub-problems. The goal feature for each sub-problem is a weighted combination of all individual goals. However, the weighted sum is one of the approaches used in *MOEA/D* algorithm. Each sub-problem is optimized by using the current information from its neighboring sub-problems at the same time. The relationship of neighborhood among all sub-problems is defined based on the distance between their aggregated weight coefficient vectors. *MOEA/D* outperforms *NSGA-II* in terms of convergence speed and computational complexity. The another multi-objective algorithm based on the concept of the *ACO* algorithm is *MOACO*, which Angus and Woodward proposed in 2009 (Angus and Woodward 2009). This algorithm is made up of several elements, including pheromone matrix, solution building, rating and solution evaluation, updating and decay, and *Pareto* archival. This will improve *MOACO* with a novel classified strategy to deliver at the nearby solutions and provides an actual balance between *exploration* capabilities and extraction.

Yu et al. (2016) proposed a priori approach which decomposes the multi-objective problem into a number of scalar problems to find the regions of interest for the decision-maker. This work was further extended to interactively find the regions of interest (Zheng et al. 2017). Yu et al. (2019) introduced the new line of preference-driven optimization method to find the knee regions instead of conventional regions of interest with respect to the preference information the decision-maker gives. In addition to the above algorithms, there are several other methods of optimization such as the Multi-Cat Swarm Optimization (*MOCSO*) (Pradhan and Panda 2012), Multi-agent Genetic Algorithm (*MAGA*) (Zhong et al. 2004), Multi-objective Artificial Bee Colony Algorithm (*MOABC*) (Hancer et al. 2015; Luo et al. 2017), and Multi-objective Flower Pollination Algorithm (*MOFPA*) (Yang et al. 2014). Although these algorithms can very effectively and efficiently approximate *Pareto's* set of optimal solutions, they are not able to solve all *NFL* theorem optimization problems (Wolpert and Macready 1997). Therefore, a new algorithm is required to be able to resolve problems that was not solved by the existing optimization techniques.

3 Proposed algorithm

In this section, firstly the brief overview of *MOSHO*, *SSA*, and *EPO* algorithms is discussed. Then, the proposed hybrid algorithm is presented by merging the features of these algorithms.

3.1 Multi-objective Spotted Hyena Optimizer (MOSHO)

MOSHO is the multi-objective variant of a newly developed *SHO* algorithm based on the natural behaviors of spotted hyenas. *SHO* focuses primarily on attacking, searching, hunting, and encircling actions of the trusted group of spotted hyenas. *SHO's* encircling behavior is described by the following equations (Dhiman and Kumar 2017):

$$\vec{X}_h = | \vec{A} \cdot \vec{C}_p(x) - \vec{C}(x) |, \quad (1)$$

$$\vec{C}(x+1) = \vec{C}_p(x) - \vec{B} \cdot \vec{X}_h, \quad (2)$$

where $\vec{X}_h$ represents the distance that spotted hyena has to cover to reach the prey. x signifies the running iteration at given instance. The position vectors corresponding to prey and spotted hyena are represented by $\vec{C}_p$ and $\vec{C}$, respectively. $\cdot$ and $||$ symbols are used for representing the multiplication vector and absolute value, respectively. In addition, $\vec{A}$ and $\vec{B}$ indicate the coefficient vectors and are computed as follows:

$$\vec{A} = 2 \cdot \vec{rnd}_1, \tag{3}$$

$$\vec{B} = 2\vec{h} \cdot \vec{rnd}_2 - \vec{h}, \tag{4}$$

$$\vec{h} = 5 - \left(Itr \times \frac{5}{Max_{Itr}} \right), \quad where \quad Itr = 0, 1, 2, \ldots, Max_{Itr}, \tag{5}$$

Here, $\vec{rnd}_1$ and $\vec{rnd}_2$ are two random vectors with values in the range of [0, 1]. By changing the values of position vectors $\vec{A}$ and $\vec{B}$, spotted hyenas can reach several different places. Furthermore, the value of $\vec{h}$ is linearly decremented from 5 to 0 over the passage of maximum number of iterations, to maintain a proper balance between exploration and exploitation.

In addition, the spotted hyenas typically live and hunt in groups of trusted mates and have the ability to recognize the prey's location. The best spotted hyena (search agent), whoever is optimum, knows where the prey and other search agents are located and makes a group of trusted friends and gathers toward the best search agent. The hunting area theoretically is defined using the following equations:

$$\vec{X}_h = | \vec{A} \cdot \vec{C}_h - \vec{C}_k |, \tag{6}$$

$$\vec{C}_k = \vec{C}_h - \vec{B} \cdot \vec{X}_h, \tag{7}$$

$$\vec{O}_h = \vec{C}_k + \vec{C}_{k+1} + \cdots + \vec{C}_{k+N}, \tag{8}$$

where $\vec{C}_h$ and $\vec{C}_k$ signify the location of best search agent and other spotted hyenas, respectively. N represents the number of spotted hyenas and is calculated as follows:

$$N = count_{nos}(\vec{C}_h, \vec{C}_{h+1}, \vec{C}_{h+2}, \ldots, (\vec{C}_h + \vec{M})), \tag{9}$$

$$\vec{C}(x+1) = \frac{\vec{O}_h}{N} \tag{10}$$

Here, $\vec{M}$ represents a random vector with values in the range of [0.5, 1], nos indicates the number of candidate solutions that are somehow close to the best optimal solution in the search space being considered, and $\vec{O}_h$ is the cluster of optimal solutions. $\vec{C}(x+1)$ determines the best possible solution and helps to update the positions of other spotted hyenas based on the best search agent location.

Spotted hyena discovery is achieved with a vector $\vec{B}$. Vector $\vec{B}$ includes random values that are either > 1 or < 1 and compels the movement of spotted hyenas away from predators. Therefore, $\vec{A}$ ubiquitous aids in discovery and keeps random values within the range of [0, 5], which serves as prey weights. The exploitation of *SHO* algorithm starts with $| \vec{B} | < 1$, where $\vec{B}$ contains random values lying in the range of $[-1, 1]$.

SHO algorithm optimization cycle starts with the generation of a set of random solutions as the initial population of spotted hyenas. The other search agents form a cluster of trusted friends toward the best search agent within this group, and further update their positions. Finally, the best positions corresponding to the search agents are retrieved as optimal solutions upon completion of the termination criteria. In 2018, the *SHO* algorithm was further expanded to generate its multi-objective version called *MOSHO* (Dhiman and Kumar 2018) by adding two new components including archive and group selection mechanism. In subsequent subsections, the brief description of those components is given.

3.1.1 Archive

Also known as an external repository, the archive refers to a storage space that contains archives of all the non-dominated solutions that have been obtained so far. It consists of two main components such as the archive controller and the grid (Coello Coello et al. 2004).

The archive controller monitors the archives when the archive is complete or when a new optimal *Pareto* solution needs to move into the archive. It compares the non-dominated solutions obtained with the current archive contents at each iteration, and decides whether or not to put a solution in that archive if the following conditions are met:

- If the archive is null, the current solution will be added to the list.
- The solution can only be held in the archive if neither the archive members nor the new solution can win one another in terms of *Pareto* dominance relationship.
- If the archive has exceeded the maximum permissible size, then adaptive grid mechanism is first invoked to re-organize objective spatial segmentation by omitting one of the most crowded segment solutions. Later, the latest solution will be put in the least crowded section to maximize *Pareto's* optimum front diversity (Ruan et al. 2017).

The other part named grid (adaptive grid) is used to distribute the *Pareto* fronts uniformly (Knowles and Corne 2000). The archive's objective function space is divided into distinct regions. When an individual put in an external population resides outside the existing grid boundaries, the grid is re-calculated and transferred to each person within it.

3.1.2 Group selection mechanism

Group selection process aims to fit the most recent solution to the existing team members. It chooses the least crowded space search section and recommends one of its non-dominated solutions for the nearby candidate solution cluster. Selection is made using roulette wheel selection method with the following probability:

$$U_k = \frac{g}{N_k} \tag{11}$$

Here, g is a constant variable with value > 1 and N_k indicates the amount of *Pareto* optimal solutions acquired within the kth row. *MOSHO* can manage the problems with many computationally low cost constraints.

3.2 SSA Algorithm

SSA is a meta-heuristic optimization algorithm recently proposed by Mirjalili et al. in 2017 (Mirjalili et al. 2017). This algorithm is based on the swarming behavior of salp swarms, which makes a chain of salp when navigating and foraging in the deep sea. The population is divided into two main groups in the salp chain, namely leaders and followers. The leader leads the entire chain from the end, while the followers chase each other. In a n-dimensional search environment, the position of the leader is modified according to the following equation:

$$y_k^1 = \begin{cases} F_k + a_1((U_k - L_k)a_2 + L_k), & a_3 \geq 0 \\ F_k - a_1((U_k - L_k)a_2 + L_k), & a_3 < 0 \end{cases} \tag{12}$$

where y_k^1 is the position of leader in the kth dimension, U_k and L_k signify the upper bound and lower bound of kth dimension, respectively, F_k shows the position of food source,and a_1, a_2, and a_3 represent the random numbers.

The position of leader is updated with reference to food source only. The coefficient a_1 is responsible for better exploration and exploitation and is defined as follows:

$$a_1 = 2e^{-\left(\frac{4x}{Max_{Itr}}\right)^2}, \tag{13}$$

where x represents the current iteration and Max_{Itr} indicates the maximum number of iterations. Furthermore, the parameters a_2 and a_3 are random numbers in range of [0, 1].

The position of followers based on the Newton's law of motion is updated using the following equation:

$$y_k^j = \frac{1}{2}AT^2 + V_0T, \ j \geq 2, \tag{14}$$

where y_k^j indicates the position of j^{th} follower, T represents the time, and V_0 signifies the initial speed. The parameter A is calculated as follows:

$$A = \frac{V_{final}}{V_0}, \quad V = \frac{y - y_0}{T}, \tag{15}$$

Taking $V_0 = 0$, Equation (22) can be written as:

$$y_k^j = \frac{1}{2}(y_k^j + y_k^{j-1}), \ j \geq 2. \tag{16}$$

3.3 Emperor Penguin Optimizer (EPO)

The main objective of modeling is to identify effective mover (Dhiman and Kumar 2018). L-shape polygon plane is considered as the shape of the huddle. After the effective mover is identified, the boundary of the huddle is again computed.

3.3.1 Generate and compute the huddle boundary

To map the huddling behavior of emperor penguins, the first thing we need to consider is their polygon-shaped grid boundary. Every penguin is surrounded by at least two penguins while huddling, and the huddling boundary is decided by the direction and speed of wind flow. Wind flow is generally faster as compared to penguins movement. The huddling boundary can be mathematically formulated as:

$$\chi = \nabla\eta, \tag{17}$$

where η represents the velocity of wind and χ indicates the gradient of η. Furthermore, the complex potential is obtained by integrating vector α with η.

$$G = \eta + i\alpha, \tag{18}$$

Here, i represents the imaginary constant and G defines the polygon plane function.

3.3.2 Profile of temperature

Emperor penguins perform huddling to conserve their energy and increase huddle temperature $T = 0$ if $X > 0.5$ and $T = 1$ if $X < 0.5$, where X is the polygon radius. This temperature measure helps to perform exploration and exploitation task among emperor penguins. The temperature is computed as:

$$T' = \left(T - \frac{Max_{itr}}{y - Max_{itr}}\right), \quad T = \begin{cases} 0, & \text{if } X > 0.5 \\ 1, & \text{if } X < 0.5 \end{cases} \tag{19}$$

where y represents the current iteration, Max_{itr} indicates the maximum count of iterations, and T defines the time required to identify optimal solution.

3.3.3 Compute the distance

After the huddling boundary is figured out, the distance between the emperor penguin is computed. The current best solution is the solution with higher fitness value than previous optimum solution. The search agents update their positions corresponding to current optimal solution. The position updation can be mathematically represented as:

$$\vec{M_{ep}} = Abs\Big(N(\vec{A}) \cdot \vec{Q(y)} - \vec{P} \cdot \vec{Q_{ep}(y)}\Big), \tag{20}$$

where $\vec{M_{ep}}$ denotes the distance and y represents the ongoing iteration. $\vec{P}$ and $\vec{A}$ helps to avoid collision among penguin. $\vec{Q}$ represents the best solution, and $\vec{Q_{ep}}$ represents the position vector of emperor penguin. $N()$ denotes the social forces that helps to identify best optimal solution. The vectors $\vec{P}$ and $\vec{A}$ are calculated as follows:

$$\vec{A} = (M \times (T' + R_{grid}(Accuracy)) \times Rand()) - T', \tag{21}$$

$$R_{grid}(Accuracy) = Abs(\vec{Q} - \vec{Q_{ep}}), \tag{22}$$

$$\vec{P} = Rand(), \tag{23}$$

where M represents the movement parameter to maintain a gap between search agents for collision avoidance. The value of parameter M is set to 2. T' defines the temperature profile around the huddle, $P_{grid}(Accuracy)$ is the polygon grid accuracy, and $Rand()$ is a random number in range of [0, 1].

The function $N()$ is calculated as follows:

$$N(\vec{A}) = \left(\sqrt{f \cdot e^{-x/l} - e^{-x}}\right)^2, \tag{24}$$

where e is the expression function. f and l are control parameters for better exploration and exploitation. The values of f and l lie in the range of [2, 3] and [1.5, 2], respectively.

3.3.4 Relocation

The best solution (mover) is utilized to update the position of emperor penguins. The selected move leads to the movement of other search agents in a search space. To find next position of a emperor penguins, the following equation is used:

$$\vec{Q_{ep}}(y+1) = \vec{Q(y)} - \vec{A} \cdot \vec{M_{ep}} \tag{25}$$

where $\vec{Q_{ep}}(y+1)$ denotes the updated position of emperor penguin.

3.4 Proposed MoSSE algorithm

3.4.1 Motivation

Taking inspirations from nature, academics and practitioners conducted extensive research to develop a collection of meta-heuristic optimization algorithms. Such nature-inspired algorithms allow researchers to modify their optimization approaches based on similar algorithms in order to improve the efficacy of the solution and to quickly obtain an optimal solution to solve the complicated problems in large projects effectively. As a result, researchers began using certain algorithms to solve specific engineering problems. But, since they depend heavily on some mathematical formulas, these algorithms expose certain shortcomings and decreases the accurate understanding. Based on this condition, researchers proposed hybrid meta-heuristic optimization algorithms because these algorithms not only avoid the above-mentioned shortcomings, but also increase population diversity and exchange of individual information within the population, thus further enhancing the ability to solve complex engineering problems. Single-solution optimizer often struggles to fulfill the need for precision when solving engineering problems due to the existence of multiple design variables and restricted environments. Therefore, hybrid algorithms prove to be the most efficient methods of achieving such goals (Zhang et al. 2019). Nevertheless, according to the No-Free Lunch Theorem (*NFL*), these hybrid algorithms cannot solve all optimization problems. *NFL* (Wolpert and Macready 1997) has scientifically proved the aforementioned claim and encourages academics to build new algorithms for optimization or improve the existing ones. This has encouraged us to project a new algorithm with the aim of solving, more efficiently, many problems with the existing multi-objective solutions that are difficult to solve. In general, the inspiration behind this work can be described as below. The researchers have encountered difficulties with regard to a proper balance between exploration and exploitation characteristics. To address these concerns, it is important to develop an optimization algorithm that maintains this balance in order to enable the proposed approach to find optimal global solutions with greater accuracy (Sree Ranjini and Murugan 2017) for real-life problems. This paper makes use of the process of group selection mechanism used in *MOSHO* to provide enhanced exploration capabilities. The reasons to prefer the actions of *MOSHO* over others are as follows:

- This process removes the elimination problem of non-dominated solutions.

- Makes sure that all *Pareto* optimum objective vectors are found in the approximation set of values.

In addition, *MOSHO* has fair search capability and therefore stronger exploration capabilities with the group relations and the spotted hyena's collective behavior. But it endured the disadvantages of slow convergence and lower population density. As a result, it suffers from its robustness, since it can remain stuck in local optima at times. By comparison, the leader and follower selection mechanism is used of *SSA* algorithm based on salps swarming behavior provides optimal solutions with better convergence. On the other hand, to make sure that the next solution is possible related to previous one, relocation method of *EPO* algorithm is employed for better positioning the search agents.

3.4.2 Explanation of MoSSE

Algorithm 1 Hybrid Algorithm (MoSSE).

Initialize the population and $Itr \leftarrow 0$
Calculate the fitness value of each search agent
Determine the non-dominated solutions and initialize the archive with these solutions
$\vec{C}_h \leftarrow$ best search agent in the archive
$\vec{O}_h \leftarrow$ group of optimal solutions with reference to $\vec{C}_h$(archive)
while $(Itr < Max_{Itr})$ **do**
  for each search agent **do**
    Update the location of current search agent
      /** *Using grouping mechanism of SHO algorithm* ***/
    Apply follower and leader approaches for updated search agents
      /** *Using leader and follower approaches of SSA algorithm* ***/
    Now, relocate these search agents
      /** *Using relocation mechanism of EPO algorithm* ***/
  end for
  Compute the fitness values of all search agents
  Determine the non-dominated solutions among the updated search agents
  Update the archive with the acquired non-dominated solutions
  if archive has reached its maximum allowable size **then**
    Invoke grid approach for omitting one individual from the most crowded archive segment
    Add new solution to the archive
  end if
  if any of recently added new solutions lies outside the search space **then**
    Re-calculate the grid to adjust the new solutions
  end if
  Compute the fitness value of each search agent
  Update $\vec{C}_h$ if there exists a better solution in comparison to previous optimal solution
  Update $\vec{O}_h$ with reference to $\vec{C}_h$(archive)
  $x \leftarrow x + 1$
end while
return archive

The *MoSSE* algorithm integrating the *MOSHO*, *SSA*, and *EPO* algorithms alters the basic structure of the *MOSHO* algorithm by adding the *SSA* algorithm's leader and follower selection function and *EPO* relocation method to improve the population update process. This integration gives *MoSSE* more versatility to reinforce the balance between diversity and convergence, as well as to find the optimal solution quickly.

The pseudocode of *MoSSE* algorithm is given in Algorithm 1. The algorithm starts with the initialization of the spotted hyena population. Then, the efficiency of every search agent is calculated using the feature fitness. A selection of non-dominated solutions is selected based on their fitness. The best search agent is then searched from the search room allocated to the external repository. The next move is to use the leader and followers selection method of the *SSA* algorithm to re-calculate and further update the positions of the search agents. After that, *EPO's* relocation method helps to escape a solution from being stuck in local optima and also increases the solution's consistency with the highest convergence rate.

Following this process, each search agent's fitness value is again determined to find the non-dominated solutions and the archive is modified with those newly obtained solutions. If the archive is complete then grid mechanism is executed to delete one of the archived solutions from the highest populated section and add the new solution to the archive. Unless the newly implemented solutions lie beyond the current grid boundaries at each iteration, then grid positions are recalculated to change those solutions. The next step is to update the best search agent if a better solution exists in the database than previous optimal solution and then update the category of spotted hyenas accordingly. The algorithm continues until the conditions for the stoppage are not met.

3.4.3 Computational complexity

MoSSE algorithm's overall time complexity is $\mathcal{O}(Max_{Itr} \times n_o \times (n_p + n_{ns}) \times M \times (S + R))$ because

1. *MoSSE* population initialization requires $\mathcal{O}(n_o \times n_p)$ time. Here, n_p reflects the total population size, and n_o indicates the total number of objectives.
2. Search agent fitness is measured in $\mathcal{O}(Max_{Itr} \times n_o \times n_p)$ time, where Max_{Itr} counts the maximum number of iterations required to simulate *MoSSE* algorithm.
3. The *MoSSE* search agents takes $\mathcal{O}(M)$ time to identify. Here, M stands for the number of spotted hyenas.
4. The selection method for the follower and the leader takes $\mathcal{O}(S)$ time and relocation method requires $\mathcal{O}(R)$ time.
5. The proposed algorithm updates the list of non-dominated solutions in $\mathcal{O}(n_o \times (n_{ns} + n_p))$ time.

In addition, the *MoSSE* algorithm's space complexity is $\mathcal{O}(n_o \times n_p)$, because space is needed only during the initialization process.

4 Experimental results and discussions

In this section, we compare the results of *MoSSE* using four performance metrics to seven well-known algorithms on ten benchmark test functions. The *MoSSE* algorithm findings are compared with seven well-known multi-objective algorithms, namely *MOPSO* (Coello Coello 2002), *NSGA-II* (Deb et al. 2002), *MOEA/D* (Zhang and Li 2007), *PESA-II* (Corne et al. 2001), *MSSA* (Mirjalili et al. 2017), *MOACO* (Angus and Woodward 2009), and *MOSHO* (Dhiman and Kumar 2018). In *MATLAB R2018b*, we implemented these algorithms and executed them in 64-bit *Microsoft Windows 10* environment with 2.40 GHz *Core i7* processor with 16 GB of Random Access Memory (*RAM*). From the literature (Katunin and Przystalka 2014), it is evident that the parameter tuning can affect the efficiency of a meta-heuristic strategy, i.e., different parameter values will produce different results. The tuning of the hyperparameters is therefore done carefully, and their values are modified accordingly. The proposed algorithm contains four such parameters and will evaluate their sensitivity by assigning different values to them. We performed experiments for three benchmark test functions, namely $UF1$, $UF3$, and $UF6$.

1. **Number of iterations**: Although we assessed the impact of number of iterations on *MoSSE* results, we ran the proposed algorithm for 100, 500, 800, and 1000 iterations. The results of experiments reported in Table 1 confirm that *MoSSE* converges with the increase in the number of iterations toward equilibrium.
2. **Number of search agents**: They also address the effect of population size on *MoSSE* behavior, taking the various sizes as 50, 80, 100, and 200 into account. Table 2 explains the statistical results of more than 30 experiments on simulation. From the table, it can be observed that with 100 population size, *MoSSE* produces better optimum solutions, and the value of objective function decreases as more population size increases.
3. **Influence of archive**: To demonstrate the impact of the archive on *MoSSE*, archive operations on $UF1$, $UF3$, and $UF6$ test functions are shown in Table 3 and Fig. 1 in the form of successive iterations. For these research issues the server size is counted as 10. From Table 3 it can be observed that the proposed algorithm will acquire the optimum values for these test functions over successive generations.
4. **Influence of selection approach**: The proposed algorithm was evaluated using tournament selection and roulette wheel selection strategies for analysis of *MoSSE* results on $UF1$, $UF3$, and $UF6$ test problems. The selection mechanism convergence analyzes for the above test problems are shown in Fig. 2. Having observed this result, it should be noted that the choice of roulette wheels is better than the tournament selection approach in terms of converging toward the optimal solution.

Table 1 Sensitivity analysis of maximum number of iterations

Iterations	Test functions		
	UF1	UF3	UF6
100	8.12E–01	4.16E–00	6.17E–01
500	3.78E–02	7.82E–01	2.71E–01
800	2.11E–02	5.71E–02	2.18E–03
1000	**1.69E–05**	**2.09E–05**	**3.78E–06**

The best results are shown in bold

Table 2 Sensitivity analysis of number of search agents

Iterations	Test functions		
	UF1	UF3	UF6
50	7.92E–02	8.80E–03	6.66E–04
80	6.14E–04	3.47E–05	2.00E–04
100	**2.71E–06**	**4.67E–07**	**4.55E–06**
200	5.22E–04	4.67E–03	4.38E–01

The best results are shown in bold

4.1 Benchmark test functions and performance metrics

To demonstrate the efficiency of proposed algorithm, we applied *MoSSE* on ten *IEEE CEC-9 (UF1–UF10)* (Zhang et al. 2008) frequently employed test functions with diverse characteristics. The four well-known performance measures are employed, namely Hypervolume (*HV*) (Zitzler and Thiele Nov 1999; Coello et al. 2010), $\Delta_p (p = 1)$ (Schutze et al. 2012; Rudolph et al. Jun 2016), Spread (Coello et al. 2010; Li and Zheng 2009), and Epsilon (ε) (Zitzler et al. April 2003; Knowles et al. 2006) to quantify the convergence and coverage (diversity) of proposed algorithm.

4.2 Evaluation on IEEE CEC-9 test suite

Table 4 lists the performance metrics attained for the *MoSSE* and the *CEC-9* test suite comprising seven aforementioned algorithms. Overall, it is worth noting that for most research problems, *MoSSE* outperforms other algorithms. In addition, *MoSSE* provides best performance indicators on fifth, sixth, eighth, and seventh *CEC-9* test cases, respectively, in terms of average and *SD* values of *HV*, Δ_p, *Spread*, and *Epsilon*. For the particular case of *UF1* test feature, *MoSSE* provides

Table 3 The archive values obtained by *MoSSE* algorithm using multiple runs

Iterations	UF1				UF3				UF6			
	Archive		Objective value		Archive		Objective value		Archive		Objective value	
	x	y	f_1	f_2	x	y	f_1	f_2	x	y	f_1	f_2
1	0.981	0.011	0.858	0.923	0.252	3.602	0.641	0.371	0.982	0.011	0.442	0.031
	0.982	0.149			0.002	5.971			0.972	0.165		
	0.944	0.281			0.002	5.978			0.942	0.281		
	0.902	0.418			0.022	5.321			0.892	0.417		
	0.821	0.533			0.023	5.328			0.833	0.523		
	0.743	0.660			0.002	5.983			0.743	0.646		
	0.637	0.753			0.083	4.106			0.636	0.743		
	0.523	0.838			0.025	5.333			0.433	0.838		
	0.403	0.913			0.244	3.608			0.403	0.893		
	0.257	0.960			0.043	5.333			0.262	0.954		
50	0.837	0.013	0.763	0.748	0.125	2.023	0.583	0.231	0.816	0.010	0.303	0.033
	0.857	0.137			0.023	3.233			0.752	0.254		
	0.803	0.111			0.025	4.020			0.873	0.243		
	0.840	0.341			0.033	4.229			0.748	0.373		
	0.733	0.411			0.027	3.003			0.733	0.337		
	0.626	0.513			0.080	4.457			0.682	0.521		
	0.547	0.430			0.122	3.083			0.751	0.416		
	0.432	0.743			0.022	3.022			0.332	0.771		
	0.402	0.823			0.211	2.213			0.392	0.836		
	0.267	0.758			0.032	4.413			0.217	0.740		
100	0.026	0.903	0.761	0.556	0.920	0.147	0.544	0.961	0.011	0.924	0.259	0.913
	0.111	0.664			0.971	0.145			0.039	0.783		
	0.221	0.536			0.948	0.293			0.031	0.700		
	0.329	0.424			0.912	0.413			0.077	0.660		
	0.442	0.347			0.830	0.543			0.201	0.467		
	0.472	0.283			0.728	0.658			0.231	0.273		
	0.604	0.213			0.582	0.787			0.431	0.173		
	0.682	0.163			0.523	0.841			0.412	−0.092		
	0.804	0.103			0.461	0.914			0.617	−0.257		
	0.892	0.063			0.250	0.967			0.812	−0.513		

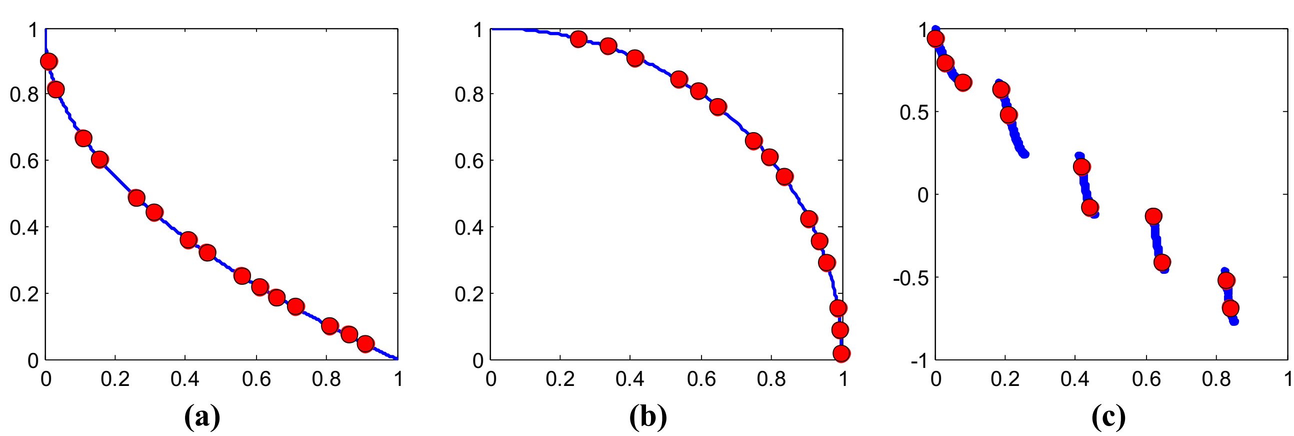

Fig. 1 Effect of archive on **a** U1, **b** UF3, and **c** UF6 test functions

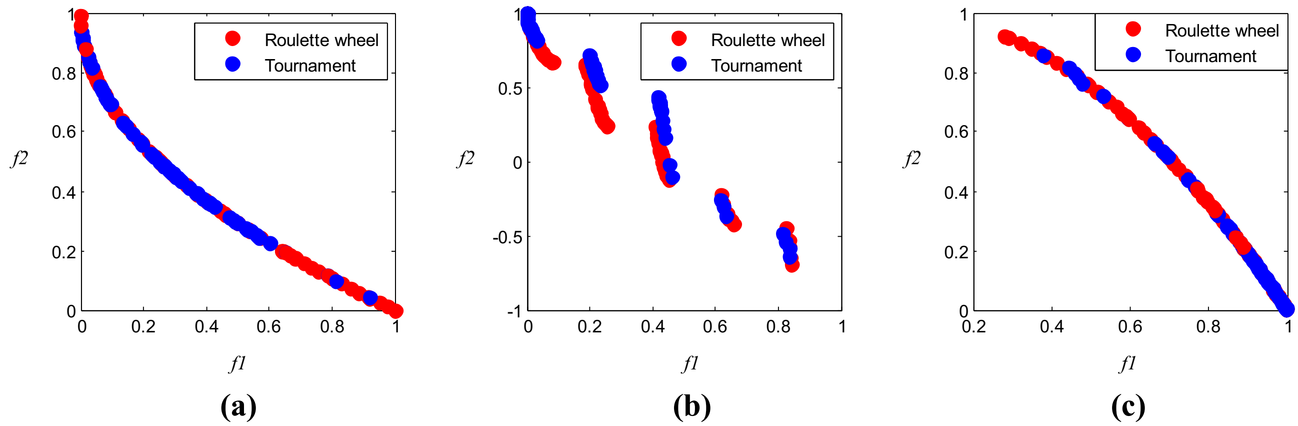

Fig. 2 Effect of selection mechanism on **a** UF1, **b** UF3, and **c** UF6 benchmark test problems

best results in terms of all four efficiency measures relative to all other successful algorithms. Subsequently, *MOSHO* performs better in terms of both *SD* and mean value. For *HV*, the average *MOEA/D* algorithm value is higher, while for *MOSHO* algorithm, the standard deviation is promising. *MoSSE* reveals great output in terms of *HV*, *Spread*, and *Epsilon* for *UF2* test feature. For *UF3* benchmark test issue, in comparison with other multi-objective methods, *MOEA/D* provides superior results in terms of the average value of Δ_p and *Epsilon* metric. But for *HV* and *Spread*, respectively, this algorithm falls behind *MSSA* and *MOPSO*. In addition, the *SD* obtained by *MOACO* and *MOPSO* is higher compared to the competitor algorithms for the performance metrics. *MoSSE* produces very positive results compared to the *MOSHO* algorithm for *UF4* and *UF5* test functions, and also outperforms it. For *UF4*, the proposed algorithm produces very promising results for all output metrics considered except for the *SD* of *Spread* metric. In comparison, *MoSSE* obtains the best *HV*, Δ_p, and *Spread* values for *UF5* test function but does not have better *Epsilon* metric values. For *UF6* and *UF7*, both *MoSSE's Spread* and *Epsilon* metrics are smaller than *MOSHO's* and substantially lower than its competitive algorithms. In addition, *MoSSE* achieves strong *HV* but does not achieve better values for *UF8* testing as these values have already been bagged by *MOEA/D*, *PESA-II*, and *NSGA-II* algorithms. In addition, the optimal solutions procured by the *Pareto* are represented in Figure 3. To track the algorithm's proposed coverage, this figure shows that *MoSSE's* optimal *Pareto* fronts are very similar to the actual optimal front of *Pareto*. The results described above show that *MoSSE* can provide impressive coverage and convergence to solve multi-objective problems.

4.3 Statistical testing

The Mann–Whitney *U* rank sum test (Mann and Whitney 1947) has been conducted on the average values of *IEEE CEC-9* benchmark test functions. Table 5 shows the results of the Mann–Whitney *U* rank sum test for statistical superiority of proposed *MoSSE* at 0.05 significance level. First, the significant difference between two data sets has been observed by Mann–Whitney *U* rank sum test. If there is no significant difference then sign '=' appears and when significant difference is observed then signs '+' or '-' appears for the case where *MoSSE* takes less or more function evaluations than the other competitor algorithms, respectively. In Table 5, '+' shows that *MoSSE* is significantly better and '-' shows that *MoSSE* is worse. As Table 5 includes 58 '+' signs out of 70 comparisons. Therefore, it can be concluded that the results of *MoSSE* is significantly effective than competitive approaches (i.e., *MOSHO*, *MOPSO*, *NSGA-II*, *MOEA/D*, *PESA-II*, *MSSA*, and *MOACO*). Overall, the results reveal that *MoSSE* outperforms other algorithms.

5 MoSSE for real-life applications

MoSSE algorithm is applied to four real-life engineering design problems to test the efficacy of the proposed algorithm, namely welded beam design, multiple-disk clutch brake design, pressure vessel design, and 25-bar truss design problems. There are different types of penalty functions to handle these multi-constraint problems. These penalty functions are static penalty, dynamic penalty, annealing penalty, adaptive penalty, co-evolutionary penalty, and death penalty (Coello Coello 2002). However, death penalty function is used to discard the infeasible solutions and does not employ the information of such solutions which are helpful to solve the dominated infeasible regions. Due to low computational cost and its simplicity, *MoSSE* algorithm is equipped with death penalty function to handle multiple constraints.

Table 4 Optimization results of multi-objective algorithms on *IEEE CEC-9* test suite

F	Performance metrics	MoSSE	MOSHO	MOPSO	NSGA-II	MOEA/D	PESA-II	MSSA	MOACO
UF1	*Hypervolume*	**2.57E–00**	4.69E–02	3.81E–01	4.00E–01	6.61E–01	3.82E–01	1.74E–01	1.45E–01
		5.20E–05	7.30E–04	2.01E–02	3.35E–03	2.81E–03	4.96E–02	1.21E–01	6.80E–02
	Δ_p	**2.15E–10**	4.07E–09	3.22E–04	2.41E–03	4.66E–04	5.41E–03	1.60E–02	2.66E–03
		6.01E–12	8.11E–11	3.10E–04	6.33E–03	1.02E–04	1.71E–03	1.67E–03	1.91E–03
	Spread	**2.14E–02**	1.27E–01	7.53E–01	2.24E+00	3.12E–01	1.41E+00	1.03E+00	1.01E+00
		1.01E–03	3.12E–02	2.41E–01	1.23E–01	1.84E–01	1.62E–01	1.04E–01	1.02E–01
	Epsilon	**3.80E–04**	5.90E–03	1.05E–01	2.24E–01	1.18E–02	1.75E–01	5.55E–01	1.82E–01
		2.22E–04	1.11E–03	3.00E–02	4.51E–02	1.16E–02	1.03E–01	1.68E–01	1.21E–01
UF2	*Hypervolume*	**2.11E–00**	1.07E–01	5.11E–01	5.11E–01	5.50E–01	4.93E–01	4.26E–01	3.15E–01
		2.23E–04	1.93E–03	1.14E–03	6.11E–03	2.88E–03	1.23E–02	2.06E–01	1.27E–01
	Δ_p	**1.11E–04**	4.13E–04	1.50E–03	4.77E–03	3.41E–04	2.62E–03	2.20E–03	2.11E–03
		2.11E–03	2.10E–04	**2.85E–05**	4.62E–04	1.74E–04	4.85E–04	1.10E–03	1.97E–03
	Spread	**1.35E–03**	4.09E–02	3.97E–01	5.03E–01	3.21E–01	6.38E–01	2.02E+00	2.15E–01
		3.41E–03	2.93E–02	1.22E–01	3.71E–02	4.50E–02	7.11E–02	2.22E–01	1.48E–01
	Epsilon	**3.33E–03**	2.95E–02	8.22E–02	2.30E–01	4.81E–02	1.78E–01	2.34E–01	3.70E–02
		5.22E–05	5.19E–04	8.72E–03	2.33E–02	1.26E–02	4.60E–02	1.51E–01	2.42E–03
UF3	*Hypervolume*	5.17E–01	5.09E–01	1.75E–00	1.13E–00	4.03E–00	1.44E–00	**7.01E–00**	2.63E–00
		4.43E–02	3.70E–02	2.50E–02	2.58E–02	2.21E–02	2.35E–02	2.74E–02	**1.20E–02**
	Δ_p	2.51E–03	2.63E–03	4.87E–03	7.95E–03	**1.05E–03**	2.53E–02	2.47E–02	1.87E–02
		2.53E–03	1.61E–03	**3.30E–04**	1.61E–03	4.00E–04	1.88E–03	1.98E–03	1.41E–03
	Spread	6.22E–01	8.02E–01	**4.23E–01**	2.31E+00	4.37E–01	2.00E+00	1.08E+00	2.47E+00
		4.33E–02	6.99E–02	3.18E–01	4.46E–02	2.04E–01	1.02E–01	5.38E–02	**1.75E–02**
	Epsilon	1.32E–01	2.93E–01	2.83E–01	2.83E–01	**1.30E–01**	3.77E–01	4.54E–01	2.95E–01
		3.35E–02	4.17E–02	**2.43E–03**	3.53E–02	4.10E–02	4.30E–02	6.47E–02	3.47E–02
UF4	*Hypervolume*	**2.11E–00**	1.00E–01	1.41E–01	1.45E–01	1.36E–01	1.41E–01	1.51E–01	1.40E–01
		1.24E–03	2.42E–03	6.70E–03	2.18E–03	1.00E–02	2.12E–03	5.51E–02	1.96E–02
	Δ_p	**2.23E–05**	4.15E–04	1.65E–03	1.30E–03	2.88E–03	1.38E–03	4.61E–03	3.98E–03
		2.35E–06	2.69E–05	1.03E–04	3.44E–05	4.68E–04	1.31E–04	2.63E–03	1.80E–04
	Spread	2.61E–01	**1.01E–01**	3.27E–01	3.12E–01	3.02E–01	7.23E–01	2.03E+00	2.75E–01
		1.57E–02	3.43E–02	5.82E–02	3.13E–02	8.45E–02	4.31E–02	3.22E–01	3.23E–02
	Epsilon	**1.00E–02**	2.12E–02	6.47E–02	4.98E–02	5.80E–02	7.34E–02	1.61E–01	2.71E–02
		1.24E–04	3.27E–03	4.28E–02	7.14E–03	8.13E–03	7.60E–03	2.11E–01	5.84E–03
UF5	*Hypervolume*	**2.70E–01**	2.92E–03	2.50E–03	1.71E–02	3.71E–02	5.21E–02	2.95E–02	2.49E–02
		1.12E–02	1.23E–02	1.56E–02	3.05E–02	3.61E–02	5.00E–02	3.68E–01	1.48E–02
	Δ_p	**1.01E–01**	1.05E–01	1.62E–01	1.88E–01	2.04E–01	1.99E–01	3.26E–00	2.41E–01
		1.00E–02	3.16E–02	1.51E–01	2.88E–02	1.90E–02	1.37E–02	2.76E–01	1.98E–02
	Spread	**1.42E–01**	2.90E–01	5.56E–01	1.25E+00	1.21E+00	1.13E+00	3.82E+00	1.20E+00
		2.36E–02	4.68E–02	7.01E–02	8.94E–02	4.27E–02	1.33E–01	6.38E–02	3.72E–02
	Epsilon	1.13E–01	**1.11E–01**	2.26E+00	6.91E–01	6.41E–01	7.80E–01	9.71E–01	4.96E–01
		1.25E–01	**1.00E–01**	4.27E–01	1.58E–01	1.42E–01	1.75E–01	1.59E–01	2.46E–01
UF6	*Hypervolume*	**3.52E–00**	4.60E–02	6.02E–02	2.95E–01	3.07E–01	2.09E–01	4.14E–01	3.88E–01
		1.23E–03	3.18E–02	2.47E–02	5.98E–02	9.23E–02	4.39E–02	7.60E–02	3.20E–02
	Δ_p	2.24E–03	4.51E–03	8.55E–03	4.67E–03	**2.13E–03**	2.92E–02	5.22E–02	7.79E–02

Table 4 continued

F	Performance metrics	MoSSE	MOSHO	MOPSO	NSGA-II	MOEA/D	PESA-II	MSSA	MOACO
		1.24E–04	1.60E–03	3.55E–03	1.50E–03	1.66E–03	2.78E–03	1.44E–03	7.28E–03
	Spread	**5.17E–01**	7.09E–01	8.78E–01	1.18E+00	1.13E+00	1.26E+00	1.11E+00	1.10E+00
		2.32E–02	4.50E–02	5.73E–02	1.23E–01	8.46E–02	2.69E–01	8.27E–02	4.40E–02
	Epsilon	**1.11E–01**	2.22E–01	4.94E–01	4.38E–01	2.54E–01	4.47E–01	7.71E–01	4.38E–01
		1.00E–01	1.20E–01	1.81E–01	1.48E–01	1.47E–01	1.90E–01	2.66E–01	1.08E–01
UF7	*Hypervolume*	2.43E–01	2.33E–01	2.94E–01	2.28E–01	3.90E–01	1.48E–01	**1.21E–00**	1.85E–01
		4.32E–04	6.16E–02	2.56E–03	7.84E–03	**2.33E–05**	7.81E–05	7.81E–03	2.98E–03
	Δ_p	1.46E–03	3.61E–03	5.37E–03	6.77E–03	**1.50E–04**	5.48E–02	2.17E–02	3.36E–02
		2.57E–04	1.15E–03	1.66E–03	1.05E–03	**1.43E–05**	1.70E–03	2.31E–03	1.74E–03
	Spread	**5.99E–02**	2.91E–01	7.51E–01	1.06E+00	2.48E–01	1.17E+00	1.00E+00	4.60E–00
		2.46E–03	2.17E–02	5.62E–02	1.10E–01	1.74E–01	7.76E–02	1.57E–02	2.57E–02
	Epsilon	**2.32E–03**	2.15E–02	1.81E–01	3.27E–01	3.41E–02	5.07E–01	7.00E–01	2.61E–01
		3.11E–04	1.78E–02	1.06E–01	1.41E–01	2.16E–02	1.18E–01	1.36E–01	1.80E–01
UF8	*Hypervolume*	**3.26E–00**	1.60E–02	2.04E–02	2.26E–01	1.80E–01	1.92E–02	1.28E–01	1.01E–01
		1.35E–03	2.12E–02	1.68E–02	3.06E–02	1.00E–01	2.31E–02	7.95E–02	1.78E–02
	Δ_p	1.56E–02	3.14E–03	1.91E–03	**1.80E–03**	2.15E–01	4.10E–03	3.86E–03	4.94E–03
		6.35E–03	6.11E–04	1.00E–04	**3.44E–05**	4.16E–03	3.85E–04	8.08E–04	1.85E–03
	Spread	3.45E–01	6.27E–01	8.15E–01	6.76E–01	**3.68E–01**	6.06E–01	7.61E–01	4.78E–01
		3.00E–01	5.45E–01	7.21E–02	**7.04E–02**	6.91E–01	1.10E–01	1.28E–01	1.57E–01
	Epsilon	5.35E–00	9.33E–01	6.68E–01	**5.23E–01**	6.00E–01	8.71E–01	7.04E–01	5.96E–01
		4.46E–00	3.17E–01	7.50E–02	1.10E–01	8.53E–01	**4.21E–02**	1.30E–01	2.65E–01
UF9	*Hypervolume*	6.77E–02	5.59E–02	5.88E–02	1.53E–01	1.75E–02	8.42E–02	**3.56E–01**	2.38E–01
		3.20E–01	6.57E–01	4.71E–02	5.16E–02	3.46E–01	2.50E–02	3.77E–02	**1.10E–02**
	Δ_p	7.49E–03	4.19E–03	1.67E–03	4.34E–03	5.91E–03	3.77E–03	**1.21E–03**	2.22E–03
		8.99E–04	6.10E–04	2.26E–04	1.50E–04	3.47E–03	2.30E–04	**1.03E–04**	1.75E–03
	Spread	**1.01E–01**	1.09E–01	7.20E–01	7.51E–01	5.62E–01	6.54E–01	6.35E–01	3.58E–01
		8.82E–02	7.97E–02	4.04E–02	5.82E–02	5.40E–02	6.44E–02	5.11E–02	**2.45E–02**
	Epsilon	**1.46E–01**	4.43E–01	7.20E–01	4.13E–01	5.55E–01	7.94E–01	3.80E–01	5.84E–01
		3.58E–02	5.93E–02	1.41E–01	5.45E–02	4.76E–02	3.15E–02	**1.91E–02**	4.67E–02
UF10	*Hypervolume*	6.46E–02	4.90E–02	**1.00E+00**	3.52E–03	5.91E–02	3.04E–02	2.37E–02	3.91E–02
		1.77E–01	1.31E–02	0.00E+00	**1.15E–02**	4.10E–01	1.96E–02	1.85E–02	2.01E–02
	Δ_p	1.41E–02	6.18E–03	2.78E–02	2.12E–03	3.00E–03	**1.82E–03**	5.51E–03	3.21E–03
		4.76E–02	4.61E–03	5.72E–03	2.01E–03	1.80E–03	3.17E–04	**2.46E–04**	1.45E–03
	Spread	**1.55E–01**	4.52E–01	5.52E–01	6.65E–01	4.20E–01	8.17E–01	1.23E+00	3.52E–01
		5.55E–01	5.53E–02	**4.06E–02**	5.27E–02	7.61E–01	8.12E–02	1.22E–01	1.85E–01
	Epsilon	**2.11E–01**	4.47E–01	1.81E+00	1.04E+00	4.32E–01	7.85E–01	7.67E–01	5.01E–01
		3.96E–01	7.90E–01	1.44E–01	1.54E–01	6.16E–01	1.06E–01	**1.02E–01**	1.84E–01

The best results are shown in bold

5.1 Welded beam design problem

Coello Coello (2000) projected the problem of welded beam, which was also regarded as a traditional benchmark concern in the field of structural optimization. This helps to minimize production costs while reducing the final deflection of welded beams that are subject to four non-linear constraints by optimizing four design parameters, namely welded joint length (l), bar thickness (b), bar height (t), and welding thickness (h). The related other constraints include load buckling, stress bending, deflection of the end beam, and strain of the shear. Figure 4 outlines the welded beam design problem along with its geometrical parameters. Table 6 lists the best *MoSSE* solutions and other meta-heuristic algorithms. The findings show that the proposed algorithm greatly reduced the cost of manufacturing along with end deflection as compared to the findings of other effective algorithms. The proposed algo-

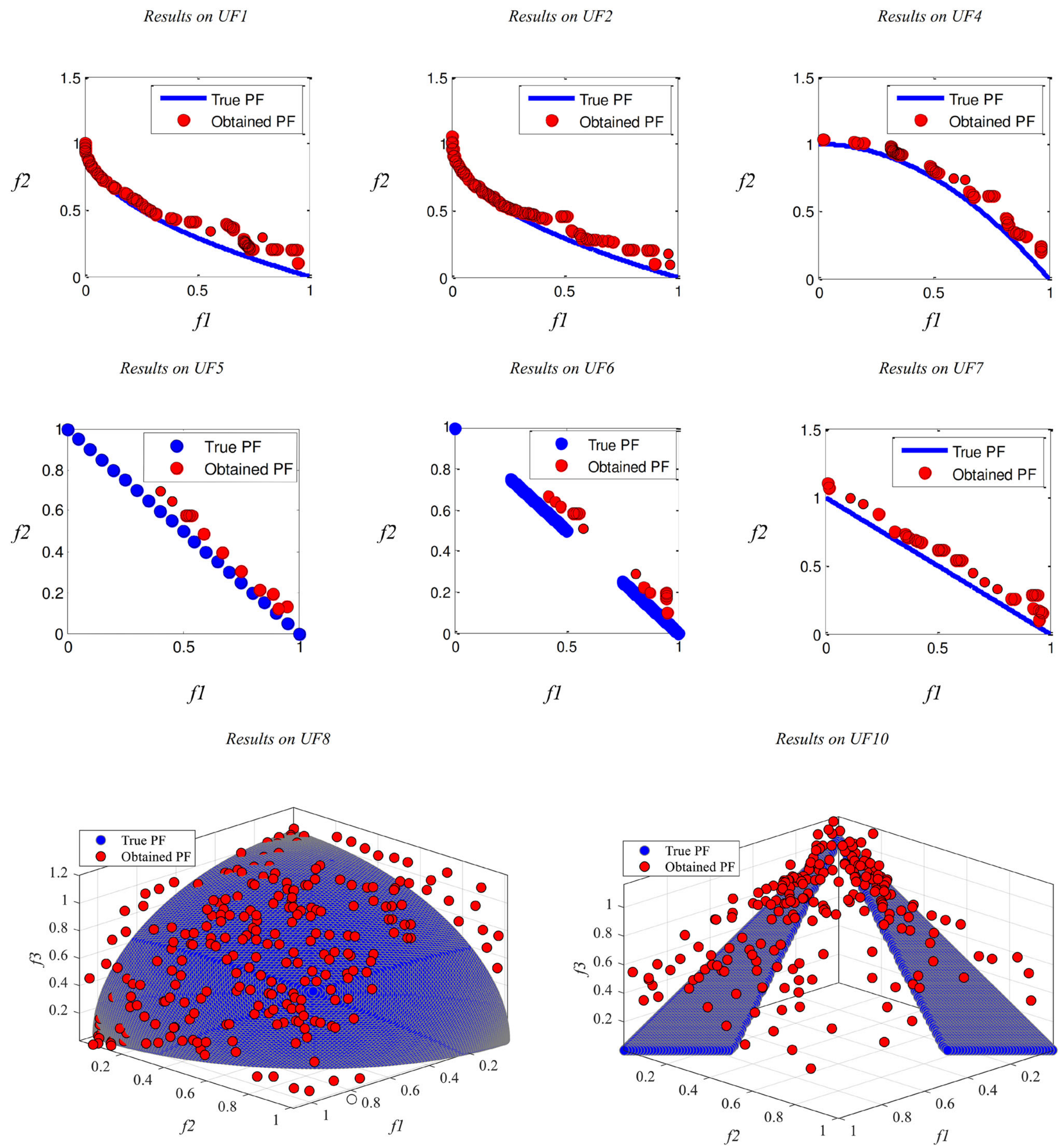

Fig. 3 The best Pareto front approximations attained by *MoSSE* on *IEEE CEC-9 (UF1–UF10)* benchmark test functions

rithm is also capable of producing a wide variety of optimal and uniformly distributed *Pareto* solutions (see Fig. 5).

5.2 Multiple-disk clutch brake design problem

Another benchmark problem in the field of multi-objective optimization with the objectives of reducing the mass and stopping time by improving the five design parameters. Such parameters of construction with outer radius, inner radius, number of surfaces, disk thickness, and actuating force are depicted in Fig. 6. The analysis of compared optimal solutions by using the proposed as well as competitor optimization algorithms is tabulated in Table 7. It is seen from this table that *MoSSE* generates best solutions and improves

Table 5 Mann–Whitney *U* rank sum test at 0.05 significance level

F	MOSHO	MOPSO	NSGA-II	MOEA/D	PESA-II	MSSA	MOACO
UF1	+	+	+	+	+	+	+
UF2	–	+	–	–	–	+	–
UF3	+	+	+	+	+	+	=
UF4	+	+	-	+	+	+	+
UF5	–	–	–	+	+	+	+
UF6	+	+	-	+	+	+	+
UF7	+	+	+	+	+	+	+
UF8	+	+	+	+	+	+	+
UF9	+	+	+	+	-	+	+
UF10	+	+	+	+	+	+	+

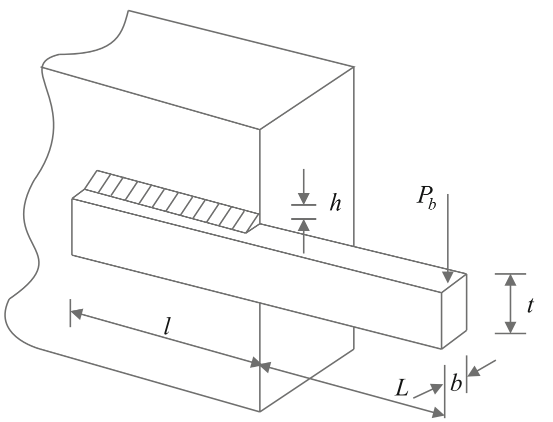

Fig. 4 Schematic depiction of the welded beam design problem

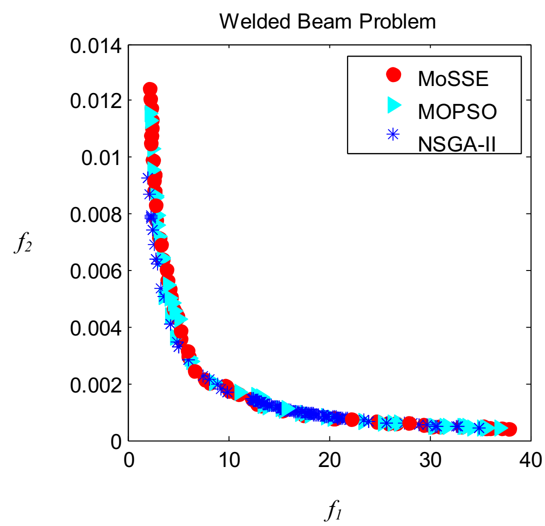

Fig. 5 Obtained Pareto optimal solutions for welded beam design problem

the overall mass and stopping time. Also, the *Pareto* optimal fronts are shown in Fig. 7. This figure reveals that the results provided by the *MoSSE* algorithm are superior than other algorithms.

5.3 Pressure vessel design problem

Kannan and Kramer (1994) suggested a design for a pressure vessel consisting of a cylindrical vessel filled with hemispheric heads at both ends, as shown in Fig. 8. The objective of this problem is to maximize storage capacity and minimize the total cost comprising material, forming and welding costs of the cylindrical vessel. It has four decision variables namely x_1, x_2, x_3 and x_4 which represent the shell thickness (T_s), head thickness (T_h), inner radius (R), and cylindrical section

Table 6 Comparison of *MoSSE* algorithm for welded beam design problem

Performance Metrics	MoSSE	MOSHO	MOPSO	NSGA-II	MOEA/D	PESA-II	MSSA	MOACO
Hypervolume	**8.46E–00**	5.18E–01	7.08E–01	6.16E–01	6.93E–01	7.81E–01	6.41E–01	7.58E–01
	3.24E–02	3.35E–01	5.92E–01	3.21E–01	5.96E–01	7.95E–01	3.86E–01	4.71E–01
Δ_p	**8.44E–03**	3.62E–02	2.35E–01	1.68E–01	2.35E–02	5.83E–01	6.91E–02	3.33E–02
	1.19E–03	4.75E–03	5.50E–02	1.82E–02	1.31E–02	5.86E–02	4.52E–02	2.49E–02
Spread	**1.28E–02**	1.42E–01	1.47E–01	8.91E–01	1.77E+00	2.71E–01	4.47E–01	2.65E–01
	2.76E–03	5.92E–02	7.28E–02	1.77E–01	2.11E+00	5.21E–02	1.17E–01	1.96E–01
Epsilon	**5.55E–03**	9.54E–03	1.03E–01	8.63E–02	3.21E–02	2.23E–01	2.62E–02	3.67E–02
	1.08E–03	1.41E–03	7.52E–02	1.35E–02	8.74E–03	8.62E–02	4.64E–03	2.01E–03

The best results are shown in bold

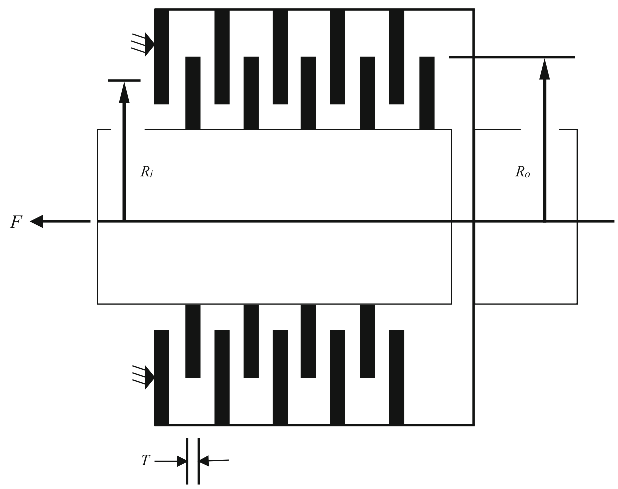

Fig. 6 Schematic representation of multiple-disk clutch brake design problem

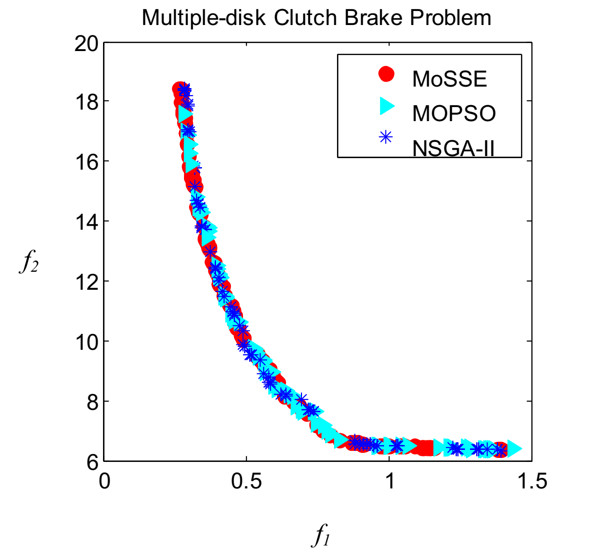

Fig. 7 Obtained Pareto solutions for multiple-disk clutch brake design problem

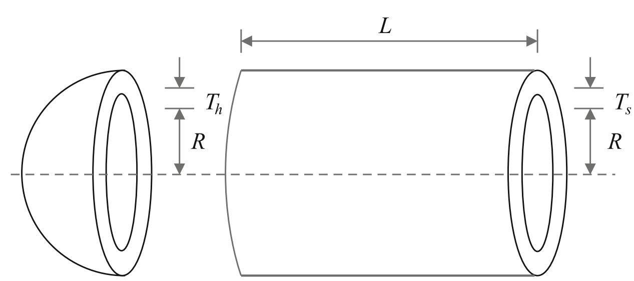

Fig. 8 Schematic depiction of pressure vessel design problem

length (L), respectively. It is also known as mixed discrete-continuous optimization problem as the first two variables are integer multiples of 0.0625 inch, whereas the later two are continuous variables.

Table 8 explains the best possible solutions obtained for problem design of pressure vessels by considered algorithms. *MoSSE* provides effective designs with maximum capacity and low cost compared to the other algorithms. This represents the efficiency and consistency of the suggested algorithm to solve the engineering problem. Further, the ideal non-dominated *Pareto* optimal fronts are shown in Fig. 9.

5.4 25-bar truss design problem

The problem is depicted in Fig. 10, which at the same time minimizes vertical displacement at node one and overall structural weight exposed to Euler buckling restriction under two load conditions. This system has 25 cross-area members, each category being divided into eight categories with maximum and minimum allowable limits (also called design variables). Tables 9 to 10 include these classification variables along with the maximum permissible tensile and compressive pressures. Further, these template variables can be picked from set: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4.

Table 11 summarizes and compares with other state-of-the-art algorithms as the best possible results. It can be found

Table 7 Comparison of *MoSSE* algorithm for multiple-disk clutch design problem

Performance metrics	MoSSE	MOSHO	MOPSO	NSGA-II	MOEA/D	PESA-II	MSSA	MOACO
Hypervolume	**5.11E+01**	4.14E–01	6.85E–01	5.04E–01	7.52E–01	8.01E–01	3.52E–01	5.41E–01
	2.23E–02	2.43E–01	4.31E–01	3.51E–01	6.62E–01	7.13E–01	6.95E–01	2.10E–01
Δ_p	**3.31E–03**	2.32E–02	4.14E–02	1.12E–01	6.42E–02	1.20E–01	8.71E–02	4.12E–02
	2.12E–04	1.57E–03	4.95E–03	1.51E–02	5.02E–02	7.81E–02	3.12E–02	3.50E–02
Spread	**7.10E–02**	1.83E–01	1.35E–01	7.62E–01	2.41E+00	7.42E–01	7.31E–01	2.30E–01
	5.02E–02	1.20E–01	1.91E–01	3.12E–01	1.34E+00	4.20E–01	3.41E–01	1.37E–01
Epsilon	**3.42E–03**	1.64E–02	1.13E–01	5.02E–02	1.38E–01	1.11E–01	1.92E–02	2.97E–02
	2.17E–04	5.42E–03	8.83E–02	1.43E–02	7.31E–02	6.31E–02	1.30E–02	1.57E–02

The best results are shown in bold

Table 8 Comparison of *MoSSE* algorithm for pressure vessel design problem

Performance Metrics	MoSSE	MOSHO	MOPSO	NSGA-II	MOEA/D	PESA-II	MSSA	MOACO
Hypervolume	**6.72E+00**	3.93E-01	6.20E-01	5.24E-01	2.02E-01	6.24E-01	6.50E-01	2.10E-01
	2.70E-02	1.92E-01	2.10E-01	2.41E-01	5.01E-01	6.00E-01	6.83E-01	1.67E-01
Δ_p	**1.00E-03**	2.84E-03	5.15E-02	3.30E-02	1.58E-02	2.90E-03	2.66E-02	1.27E-02
	2.10E-03	3.42E-03	1.18E–02	3.44E–03	4.97E–02	2.21E–02	8.91E–02	1.62E–02
Spread	**1.41E–01**	1.78E–01	6.00E–01	3.31E–01	2.32E+00	7.44E–01	8.62E–01	3.17E–01
	1.02E–01	1.29E–01	1.55E–01	2.61E–01	1.03E+00	5.37E–01	5.34E–01	1.88E–01
Epsilon	**6.77E–03**	2.27E–02	2.48E–01	8.51E–02	2.46E–01	1.41E–01	4.29E–02	2.57E–02
	8.49E–04	4.43E–03	7.42E–02	6.83E–02	1.34E–01	2.12E–01	2.81E–02	2.17E–02

The best results are shown in bold

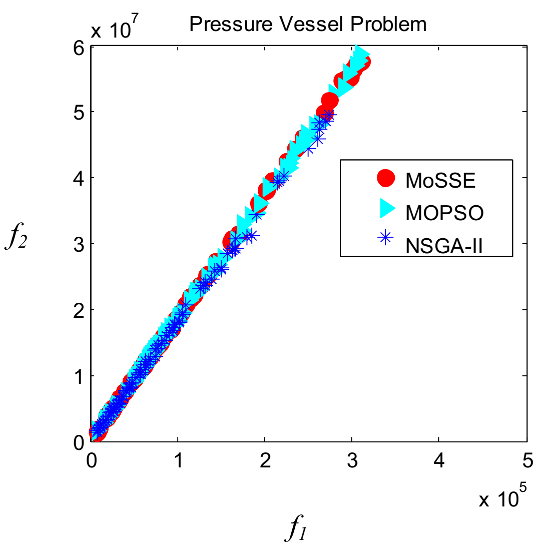

Fig. 9 Obtained Pareto optimal solutions for pressure vessel design problem

that when it comes to certain performance metrics, *MoSSE* competes other algorithms. However, by inspecting Table 11, it can be seen the *MoSSE* generated better truss design as compared to other competitor algorithms. The obtained best *Pareto* optimal solutions are given in Fig. 11 which shows that *MoSSE* convergence is better than other competitor algorithms.

6 Conclusions

This paper introduces a new multi-objective hybrid algorithm called *MoSSE* to solve real-life problems of engineering design. *MoSSE* incorporates the quick search and learning method of *MOSHO* with the leading and follow-up selection process of *SSA* and relocation method of *EPO* in order to achieve optimal global solutions. Such integration has helped maintain the suitable balance between the *exploration*

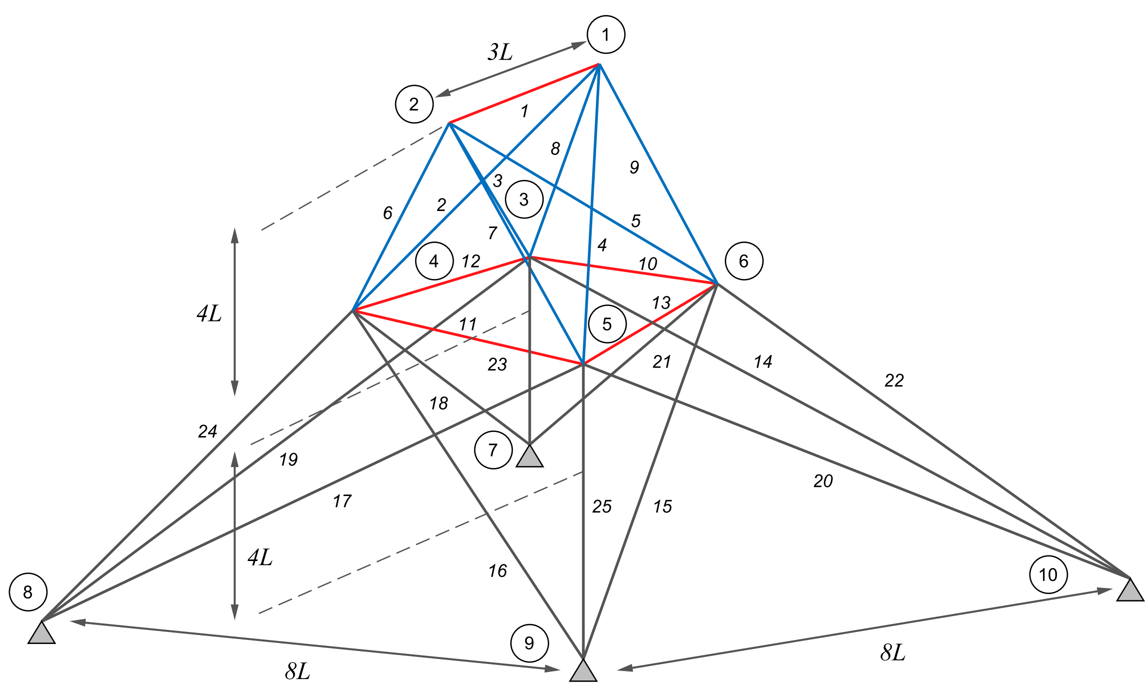

Fig. 10 Schematic representation of 25-bar truss design problem

Table 9 Load conditions of issue with 25-bar truss configuration

Node	Case 1			Case 2		
	$P_x Kips(kN)$	$P_y Kips(kN)$	$P_z Kips(kN)$	$P_x Kips(kN)$	$P_y Kips(kN)$	$P_z Kips(kN)$
1	0.0	20.0 (89)	−5.0 (22.25)	1.0 (4.45)	10.0 (44.5)	−5.0 (22.25)
2	0.0	−20.0 (89)	−5.0 (22.25)	0.0	10.0 (44.5)	−5.0 (22.25)
3	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0
6	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0

Table 10 The grouping of 25-bar truss members and their corresponding allowable stresses

Element group	Compressive stress (Kpsi)	Tensile stress (Kpsi)
Group 1: A_1	35.092 (241.96)	40.0 (275.80)
Group 2: A_2, A_3, A_4, A_5	11.590 (79.913)	40.0 (275.80)
Group 3: A_6, A_7, A_8, A_9	17.305 (119.31)	40.0 (275.80)
Group 4: A_{10}, A_{11}	35.092 (241.96)	40.0 (275.80)
Group 5: A_{12}, A_{13}	35.092 (241.96)	40.0 (275.80)
Group 6: A_{14}, A_{15}, A_{17}	6.759 (46.603)	40.0 (275.80)
Group 7: A_{18}, A_{19}, A_{20}, A_{21}	6.959 (47.982)	40.0 (275.80)
Group 8: A_{22}, A_{23}, A_{24}, A_{25}	11.082 (76.410)	40.0 (275.80)

Table 11 Comparison of *MoSSE* algorithm for 25-bar truss design problem

Performance Metrics	MoSSE	MOSHO	MOPSO	NSGA-II	MOEA/D	PESA-II	MSSA	MOACO
Hypervolume	**7.96E–00**	3.46E–00	7.57E–01	3.35E–01	6.46E–01	3.03E–01	5.74E–01	2.04E–01
	1.03E–01	1.28E–01	4.79E–01	1.42E–01	4.02E–01	1.72E–01	4.85E–01	1.32E–01
Δ_p	**1.74E–03**	2.26E–02	1.56E–01	2.79E–02	1.02E–01	1.68E–01	2.27E–02	4.38E–02
	2.49E–03	3.47E–02	1.72E–01	1.47E–02	1.34E–01	3.02E–02	1.86E–02	2.59E–02
Spread	**6.99E–02**	1.74E–01	2.59E–01	3.32E–01	1.49E–01	4.07E–01	4.38E–01	2.34E–01
	3.71E–03	3.59E–02	2.43E–01	2.27E–01	1.18E–01	3.04E–01	1.86E–01	1.02E–01
Epsilon	**6.60E–03**	1.98E–02	1.92E–01	1.82E–01	1.56E–01	2.87E–02	1.42E–01	2.66E–01
	4.13E–03	1.72E–01	2.02E–02	1.44E–01	1.82E–02	1.76E–02	1.02E–01	1.58E–01

The best results are shown in bold

and *exploitation* of the algorithm in the searching process. The efficiency of proposed algorithm is tested by using the ten standard *IEEE CEC-9* test functions, and the obtained results are compared with seven multi-objective algorithms. The result shows that the *MoSSE* algorithm provides a very good results as compared to other meta-heuristic algorithms and outperformed them in terms of achieving more accurate approximations of *Pareto's* optimal, high-convergence solutions, which are distributed equally. Additionally, the proposed algorithm's ability to solve real-life problems is also demonstrated by evaluating its efficacy on four engineering design problems. The empirical comparisons with well-established optimization algorithms demonstrate the effectiveness of *MoSSE* in the solution of real-life engineering design problems. The proposed algorithm will be used in future, especially in the cloud computing and medical engineering domains, to solve complex the problems.

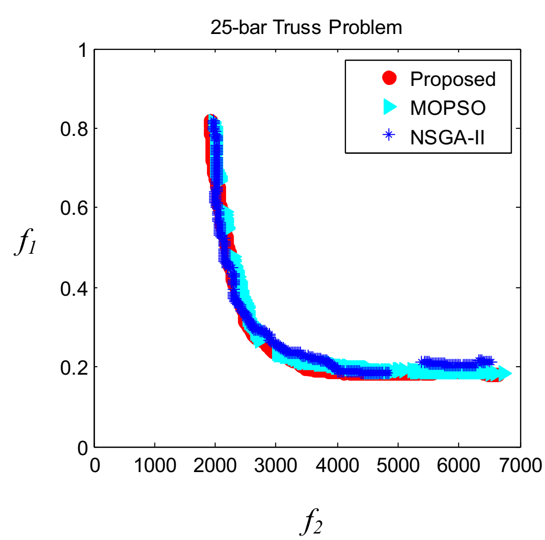

Fig. 11 Obtained Pareto optimal solutions for 25-bar truss design problem

Acknowledgements The first and corresponding author **Dr. Gaurav Dhiman** would like to thank his parents for their divine blessings on him.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Angus D, Woodward C (2009) Multiple objective ant colony optimisation. Swarm Intell 3(1):69–85

Bandarua S, Debb K (2016) Metaheuristic techniques. Decis Sci Theory Pract 693–750

Branke J, Deb K, Dierolf H, Osswald M (2004) Finding knees in multi-objective optimization. In International conference on parallel problem solving from nature. Springer, New York, pp 722–731

Chandrawat RK, Kumar R, Garg BP, Dhiman G, Kumar S (2017) An analysis of modeling and optimization production cost through fuzzy linear programming problem with symmetric and right angle triangular fuzzy number. In: Proceedings of sixth international conference on soft computing for problem solving, Springer, New York, pp 197–211

Chen M, Hammami O (2015) A system engineering conception of multi-objective optimization for multi-physics system. In: Multiphysics modelling and simulation for systems design and monitoring. Springer, pp 299–306

Chiandussi G, Codegone M, Ferrero S, Varesio FE (2012) Comparison of multi-objective optimization methodologies for engineering applications. Comput Math Appl 63(5):912–942

Coello Coello CA, Lechuga MS (2002) Mopso: a proposal for multiple objective particle swarm optimization. In: Proc., Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on, pp 1051–1056

Coello Coello CA (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127

Coello Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191(11–12):1245–1287

Coello Coello CA, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279

Coello Coello CA, Lamont GB, Van Veldhuizen DA et al (2007) Evolutionary algorithms for solving multi-objective problems, vol 5. Springer, New York

Coello CA, Coello CD, Jourdan L (2010) Multi-objective combinatorial optimization: problematic and context. Springer, Berlin, pp 1–21

Corne David W, Jerram Nick R, Knowles Joshua D, Oates Martin J (2001) Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation. Morgan Kaufmann Publishers Inc, pp 283–290

Corne David W, Knowles Joshua D, Oates Martin J (2000) The pareto envelope-based selection algorithm for multiobjective optimization. In: International conference on parallel problem solving from nature. Springer, pp 839–848

Dai C, Lei X (2019) A multiobjective brain storm optimization algorithm based on decomposition. Complexity 2019

Deb K, Pratap A, Agarwal S, Meyarivan TAMT (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197

Dehghani M, Montazeri Z, Malik OP, Dhiman G, Kumar V (2019) Bosa: binary orientation search algorithm. Int J Innovat Technol Explor Eng 9:5306–5310

Dehghani M, Montazeri Z, Malik OP, Kamal Al-Haddad, Guerrero Josep M, Dhiman G (2020) A new methodology called dice game optimizer for capacitor placement in distribution systems. Electr Eng Electromech 1:61–64

Dhiman G (2019) Esa: a hybrid bio-inspired metaheuristic optimization approach for engineering problems. Eng Comput 1–31

Dhiman G (2019) Moshepo: a hybrid multi-objective approach to solve economic load dispatch and micro grid problems. Appl Intell 1–19

Dhiman G (2019) Multi-objective metaheuristic approaches for data clustering in engineering application (s). PhD thesis

Dhiman G, Kaur A (2018) Optimizing the design of airfoil and optical buffer problems using spotted hyena optimizer. Designs 2(3):28

Dhiman G, Kaur A (2019) Stoa: A bio-inspired based optimization algorithm for industrial engineering problems. Eng Appl Artif Intell 82:148–174

Dhiman G, Kumar V (2017) Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Adv Eng Softw 114:48–70

Dhiman G, Kumar V (2018) Emperor penguin optimizer: a bio-inspired algorithm for engineering problems. Knowl Based Syst 159:20–50

Dhiman G, Kumar V (2019) Knrvea: a hybrid evolutionary algorithm based on knee points and reference vector adaptation strategies for many-objective optimization. Appl Intell 49(7):2434–2460

Dhiman G, Kumar V (2019) Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems. Knowl Based Syst 165:169–196

Dhiman G, Garg M, Nagar A, Kumar V, Dehghani M (2020) A novel algorithm for global optimization: rat swarm optimizer. J Ambient Intell Hum Comput

Dhiman G, Guo S, Kaur S (2018) Ed-sho: A framework for solving nonlinear economic load power dispatch problem using spotted hyena optimizer. Modern Phys Lett A 33(40)

Dhiman G, Kaur A (2017) Spotted hyena optimizer for solving engineering design problems. In: 2017 international conference on machine learning and data science (MLDS), IEEE, pp 114–119

Dhiman G, Kaur A (2019) A hybrid algorithm based on particle swarm and spotted hyena optimizer for global optimization. In: Soft Computing for Problem Solving. Springer, pp 599–615

Dhiman G, Kaur Ap (2020) Hkn-rvea: a novel many-objective evolutionary algorithm for car side impact bar crashworthiness problem. Int J Vehicle Des

Dhiman G, Kumar V (2018) Astrophysics inspired multi-objective approach for automatic clustering and feature selection in real-life environment. Modern Phys Lett B 32(31)

Dhiman G, Kumar V (2018) Multi-objective spotted hyena optimizer: a multi-objective optimization algorithm for engineering problems. Knowl Based Syst

Dhiman G, Kumar V (2019) Spotted hyena optimizer for solving complex and non-linear constrained engineering problems. In: Harmony search and nature inspired optimization algorithms, Springer, New York, pp 857–867

Dhiman G, Singh P, Kaur H, Maini R (2019) Dhiman: A novel algorithm for economic dispatch problem based on optimization method using Monte Carlo simulation and astrophysics concepts. Modern Phys Lett A 34(04)

Dhiman G, Soni M, Pandey Hari M, Slowik A, Kaur H (2020) A novel hybrid evolutionary algorithm based on hypervolume indicator and reference vector adaptation strategies for many-objective optimization. Eng Comput

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on. IEEE, pp 39–43

Garg M, Dhiman G (2020) Deep convolution neural network approach for defect inspection of textured surfaces. J Inst Electron Comput 2:28–38

Garg M, Dhiman G (2020) A novel content based image retrieval approach for classification using glcm features and texture fused lbp variants. Neural Comput Appl

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), pp 2420–2427

Hou Z, Yang S, Zou J, Zheng J, Yu G, Ruan G (2018) A performance indicator for reference-point-based multiobjective evolutionary optimization. In: 2018 IEEE symposium series on computational intelligence (SSCI), IEEE, pp 1571–1578

Jianjie H, Yu G, Zheng J, Zou J (2017) A preference-based multi-objective evolutionary algorithm using preference selection radius. Soft Comput 21(17):5025–5051

Kannan BK, Kramer SN (1994) An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J Mech Des 116(2):405–411

Katunin A, Przystalka P (2014) Meta-optimization method for wavelet-based damage identification in composite structures. In: 2014 Federated conference on computer science and information systems, IEEE, pp 429–438

Kaur A, Dhiman G (2019) A review on search-based tools and techniques to identify bad code smells in object-oriented systems. In: Harmony search and nature inspired optimization algorithms. Springer, pp 909–921

Kaur A, Kaur S, Dhiman G (2018) A quantum method for dynamic nonlinear programming technique using schrödinger equation and monte carlo approach. Modern Phys Lett B 32(30)

Kaveh A, Ghazaan M Ilchi (2019) A new vps-based algorithm for multi-objective optimization problems. Eng Comput 1–12

Knowles JD, Thiele L, Zitzler E (2006) A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK-Report 214

Knowles JD, Corne DW (2000) Approximating the nondominated front using the pareto archived evolution strategy. Evol Comput 8(2):149–172

Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: a tutorial. Reliab Eng Syst Safe 91(9):992–1007

Li M, Zheng J (2009) Spread assessment for evolutionary multi-objective optimization. Springer, Berlin, pp 216–230

Luh G-C, Chueh C-H (2004) Multi-objective optimal design of truss structure with immune algorithm. Comput Struct 82(11–12):829–844

Luo J, Liu Q, Yang Y, Li X, Chen M, Cao W (2017) An artificial bee colony algorithm for multi-objective optimisation. Appl Soft Comput 50:235–251

Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18(1):50–60

Mansoor U, Kessentini M, Maxim BR, Deb K (2017) Multi-objective code-smells detection using good and bad design examples. Software Qual J 25(2):529–552

Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim 26(6):369–395

Mirjalili S, Saremi S, Mirjalili SM, Coelho LS (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119

Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191

Pradhan PM, Panda G (2012) Solving multiobjective problems using cat swarm optimization. Expert Syst Appl 39(3):2956–2964

Ruan G, Yu G, Zheng J, Zou J, Yang S (2017) The effect of diversity maintenance on prediction in dynamic multi-objective optimization. Appl Soft Comput 58:631–647

Rudolph G, Schütze O, Grimme C, Domínguez-Medina C, Trautmann H (2016) Optimal averaged hausdorff archives for bi-objective problems: theoretical and numerical results. Comput Optim Appl 64(2):589–618

Satnam KLK, Awasthi ALS, Dhiman G (2020) Tunicate swarm algorithm: a new bio-inspired based metaheuristic paradigm for global optimization. Eng Appl Artif Intell

Schutze O, Esquivel X, Lara A, Coello Coello CA (2012) Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. IEEE Trans Evol Comput 16(4):504–522

Singh P, Dhiman G (2018) Uncertainty representation using fuzzy-entropy approach: special application in remotely sensed high-resolution satellite images (rshrsis). Appl Soft Comput 72:121–139

Singh P, Dhiman G (2018) A hybrid fuzzy time series forecasting model based on granular computing and bio-inspired optimization approaches. J Comput Sci 27:370–385

Singh P, Dhiman G (2017) A fuzzy-lp approach in time series forecasting. In: International conference on pattern recognition and machine intelligence, Springer, pp 243–253

Singh P, Dhiman G, Guo S, Maini R, Kaur H, Kaur A, Kaur H, Singh J, Singh N (2019) A hybrid fuzzy quantum time series and linear programming model: special application on taiex index dataset. Modern Phys Lett A 34(25)

Singh P, Dhiman G, Kaur A (2018) A quantum approach for time series data based on graph and schrödinger equations methods. Modern Phys Lett A 33(35)

Singh P, Rabadiya K, Dhiman G (2018) A four-way decision-making system for the Indian summer monsoon rainfall. Mod Phys Lett B 32(25)

Sree Ranjini KS, Murugan S (2017) Memory based hybrid dragonfly algorithm for numerical optimization problems. Expert Syst Appl 83:63–78

Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248

Verma S, Kaur S, Dhiman G, Kaur A (2018) Design of a novel energy efficient routing framework for wireless nanosensor networks. In: 2018 1st International Conference on Secure Cyber Computing and Communication (ICSCCC). IEEE, pp 532–536

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Yang X-S, Karamanoglu M, He X (2014) Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 46(9):1222–1237

Yang Y, Peng S, Zhu L, Zhang D, Qiu Z, Yuan H, Xian L (2019) A modified multi-objective self-adaptive differential evolution algorithm and its application on optimization design of the nuclear power system. Science and Technology of Nuclear Installations 2019

Yu G, Zheng J, Shen R, Li M (2016) Decomposing the user-preference in multiobjective optimization. Soft Comput 20(10):4005–4021

Yu G, Yaochu J, Markus O (2019) Benchmark problems and performance indicators for search of knee points in multiobjective optimization. IEEE Trans Cybern

Yu G, Zheng J, Li X (2015) An improved performance metric for multi-objective evolutionary algorithms with user preferences. In: 2015 IEEE congress on evolutionary computation (CEC), IEEE, pp 908–915

Zhang Q, Li H (2007) Moea/d: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6): 712–731

Zhang Z, Ding S, Jia W (2019) A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems. Eng Appl Artif Intell 85:254–268

Zhang Q, Zhou A, Zhao S, Suganthan PN, Liu W, Tiwari S (2008) Multiobjective optimization test instances for the cec 2009 special session and competition. University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report, 264

Zheng J, Yu G, Zhu Q, Li X, Zou J (2017) On decomposition methods in interactive user-preference based optimization. Appl Soft Comput 52:952–973

Zhong W, Liu J, Xue M, Jiao L (2004) A multiagent genetic algorithm for global numerical optimization. IEEE Trans Syst Man Cybern Part B (Cybern) 34(2):1128–1141

Zhou A, Bo-Yang Q, Li H, Zhao S-Z, Suganthan PN, Zhang Q (2011) Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evolut Comput 1(1):32–49

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3(4):257–271

Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. Trans Evol Comp 7(2):117–132

Zitzler E, Laumanns M, Thiele L (2001) Spea 2: improving the strength pareto evolutionary algorithm. TIK-report 103

Zitzler E, Thiele L (1998) An evolutionary algorithm for multiobjective optimization: the strength pareto approach. TIK-report 43