Knowledge-Based Systems 191 (2020) 105190

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Equilibrium optimizer: A novel optimization algorithm☆

Afshin Faramarzi [a,*], Mohammad Heidarinejad [a], Brent Stephens [a], Seyedali Mirjalili [b,1]

[a] Department of Civil, Architectural, and Environmental Engineering, Illinois Institute of Technology, Chicago, IL, USA
[b] Center for Artificial Intelligence Research and Optimization, Torrens University Australia, Fortitude Valley, Brisbane, QLD 4006, Australia

ARTICLE INFO

Article history:
Received 14 June 2019
Received in revised form 28 October 2019
Accepted 3 November 2019
Available online 6 November 2019

Keywords:
Optimization
Metaheuristic
Genetic algorithm
Particle Swarm Optimization
Physics-based

ABSTRACT

This paper presents a novel, optimization algorithm called Equilibrium Optimizer (EO), inspired by control volume mass balance models used to estimate both dynamic and equilibrium states. In EO, each particle (solution) with its concentration (position) acts as a search agent. The search agents randomly update their concentration with respect to best-so-far solutions, namely equilibrium candidates, to finally reach to the equilibrium state (optimal result). A well-defined "generation rate" term is proved to invigorate EO's ability in exploration, exploitation, and local minima avoidance. The proposed algorithm is benchmarked with 58 unimodal, multimodal, and composition functions and three engineering application problems. Results of EO are compared to three categories of existing optimization methods, including: (i) the most well-known meta-heuristics, including Genetic Algorithm (GA), Particle Swarm Optimization (PSO); (ii) recently developed algorithms, including Grey Wolf Optimizer (GWO), Gravitational Search Algorithm (GSA), and Salp Swarm Algorithm (SSA); and (iii) high performance optimizers, including CMA-ES, SHADE, and LSHADE-SPACMA. Using average rank of Friedman test, for all 58 mathematical functions EO is able to outperform PSO, GWO, GA, GSA, SSA, and CMA-ES by 60%, 69%, 94%, 96%, 77%, and 64%, respectively, while it is outperformed by SHADE and LSHADE-SPACMA by 24% and 27%, respectively. The Bonferroni–Dunn and Holm's tests for all functions showed that EO is significantly a better algorithm than PSO, GWO, GA, GSA, SSA and CMA-ES while its performance is statistically similar to SHADE and LSHADE-SPACMA. The source code of EO is publicly availabe at https://github.com/afshinfaramarzi/Equilibrium-Optimizer, http://built-envi.com/portfolio/equilibrium-optimizer/ and http://www.alimirjalili.com/SourceCodes/EOcode.zip.

1. Introduction

There are two major categories for mathematical optimization methods: (1) deterministic and (2) stochastic. Linear and non-linear programming [1,2] are some of the most commonly used deterministic methods, characterized by using the gradient information of the problem to search the space and find the solution. Although these methods are efficient for problems with linear search spaces (unimodal), they are prone to local optima entrapment when applying to problems with non-linear search spaces, including real-world non-convex problems [3,4]. To combat this issue one might start from a different initial design, modify, or hybridize the algorithm [5].

Another alternative to these conventional methods is stochastic methods, which, similar to the meta-heuristic algorithms, generate and use random variables. These algorithms are used to globally search the domain to find the global or near global optimal results. Advantages of meta-heuristics include their simplicity, independency to the problem, flexibility, and gradient-free nature [6]. Common sources of inspiration in the development of meta-heuristics are from physical phenomena, animal behavior, or evolutionary concepts. In addition, meta-heuristics are independent of the nature of the problem, meaning they do not need derivative information of the problem since they use a stochastic approach. This is in contrast to mathematical programming that typically requires detailed knowledge about the mathematical problem [5]. This independency to the nature of the problem renders them a suitable tool for finding optimal solutions for a given optimization problem without being concerned about the nonlinearity types of the problem's search space and its constraints. Another bonus is their flexibility, which allows them to solve any kind of optimization problem without major changes in the algorithms' structure. They treat the problem as a black box with input and output states, and this feature empowers them as a potential candidate for a user-friendly optimizer. In addition,

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.105190.

* Corresponding author.
E-mail address: afaramar@hawk.iit.edu (A. Faramarzi).

[1] www.alimirjalili.com.

https://doi.org/10.1016/j.knosys.2019.105190
0950-7051/

in contrast to the deterministic nature of mathematical methods, they mostly benefit from stochastic operators. Consequently, the probability of entrapment in local optima is reduced compared to conventional deterministic methods. This characteristic also renders them independent to the initial guess of solutions. Due to their ability to globally explore the search space in a reasonable amount of time as well as independency to the problem's nature, these methods become more popular and received significant attention in recent years.

Genetic Algorithms (GA) [7], Particle Swarm Optimization (PSO) [8], Simulated Annealing (SA) [9], and Ant Colony Optimization (ACO) [10] are among some of the most conventional meta-heuristics approaches. Although each of them belongs to different classes of meta-heuristics, many researchers in different areas have evaluated their performance. A common approach to assessing novel optimization algorithms is to demonstrate their competitiveness against conventional methods in solving optimization problems. It is worth mentioning that it is impossible to find an algorithm to reach to the global optimum for all class of problems. If an algorithm is tuned for an elevated performance over one class of problems, it is offset by the performance over another class.

2. Meta-heuristics algorithms

There are two important features in meta-heuristic optimization algorithms: (1) exploration and (2) exploitation. Exploration is the ability to globally search the space. This ability is associated with escaping from local optima and preventing local optima stagnation. Conversely, exploitation is the ability to locally search around promising solutions in an effort to increase their quality. Good performance is achieved by a suitable tradeoff between these two features. All population-based algorithms use these features but with different operators and mechanisms.

Based on the source of inspiration, meta-heuristics are mainly divided into four classes of: (i) evolutionary algorithms, (ii) swarm intelligence, and (iii) physics-based, and (iv) human-based methods. Evolutionary algorithms mimic rules in natural evolution. They use operators inspired by biology, such as cross-over and mutation. The most widely used evolutionary algorithm is GA, which is inspired by Darwinian evolutionary theory. GA uses the cross-over concept to produce improved solutions, called offspring, based on some fitted solutions, defined as parents. Cross-over, which naturally occurs in nature and helps maintain diversity in ecosystems; or in this sense, to explore the domain. Mutations cause the offspring to have characteristics different from their parents. This operator in GA is aimed at local search and exploitation of results. Some solutions and their dimensions experience mutation, defined by a function, and selected by a parameter such as mutation probability and percentage. Differential evolution [11], evolutionary programming [12] and evolution strategy [13] are other examples of this class.

Swarm intelligence is another class of meta-heuristics, which imitates the social behavior of animals in groups (i.e., flocks, herds, or schools). The main feature of this class is sharing of collective information of all individuals during the optimization process. The most widely used algorithm in this class is PSO, developed by Kennedy and Eberhart [8]. PSO simulates the behavior of birds flying together in flocks. PSO considers some candidates (particles) that fly over the search space in an effort to find improved solutions. During the search, they all follow the best solutions in their paths. The particles trace this path by considering their own position of best solutions, known as 'pbest', along with the best solution obtained so far, called 'gbest'. Other methods of this class include: Ant Colony Optimization [10], Cuckoo Search [14], Grey Wolf Optimizer [6], Salp Swarm Algorithm [15], and Dolphin Echolocation [16].

The third class of optimization algorithms are physics-based. These algorithms originate from physical laws in nature, and typically characterize the interaction of search agents according to governing rules rooted in physical processes. One of the most widely used algorithms in this class is Simulated Annealing [9], which uses thermodynamics laws applied to heating and then controlled cooling of a material to increase the size of its crystals. Gravitational Search Algorithm [17] employs Newton's gravitational laws between masses and their interactions to update the position toward the optimum point. Charged System Search [18] takes advantage of combining rules of physics (Coulomb's law of electrostatics) and mechanics (Newtonian laws of mechanics) to perform the optimization.

The final class of optimization is human-based algorithm, which is inspired by human interactions and human behavior in societies. For example, Imperialist Competitive Algorithm (ICA) [19] is based on the human socio-political evolution process. The populations (countries) in ICA are divided into two groups: colonies and imperialists states. The core of this algorithm is based on the competition among imperialists to take control of the colonies. In the competition, weak empires collapse and there will be only one imperialist that takes possession of all colonies. Another human-based algorithm is the Teaching-Learning-Based Optimization (TLBO), which is inspired by the influence of a teacher on learners [20]. The population in this method is divided into two parts: the "teacher phase", meaning learning from the teacher, and the "learner phase", meaning learning by interacting with other learners. These phases are consequently iterated to produce better results until convergence is achieved.

This paper develops a novel algorithm named Equilibrium Optimizer (EO), inspired by physics-based dynamic source and sink models used to estimate equilibrium states. EO falls into the third class of optimization algorithms, as it originates from physical laws in nature.

3. Equilibrium optimizer

This section presents the inspiration, mathematical model, and algorithm of the Equilibrium Optimizer (EO).

3.1. Inspiration

The inspiration for the EO approach is a simple well-mixed dynamic mass balance on a control volume, in which a mass balance equation is used to describe the concentration of a nonreactive constituent in a control volume as a function of its various source and sink mechanisms. The mass balance equation provides the underlying physics for the conservation of mass entering, leaving, and generated in a control volume. A first-order ordinary differential equation expressing the generic mass-balance equation [21], in which the change in mass in time is equal to the amount of mass that enters the system plus the amount being generated inside minus the amount that leaves the system, is described as:

$$V\frac{dC}{dt} = QC_{eq} - QC + G \tag{1}$$

C is the concentration inside the control volume (V), $V\frac{dC}{dt}$ is the rate of change of mass in the control volume, Q is the volumetric flow rate into and out of the control volume, C_{eq} represents the concentration at an equilibrium state in which there is no generation inside the control volume, and G is the mass generation rate inside the control volume. When $V\frac{dC}{dt}$ reaches to zero, a steady equilibrium state is reached. A rearrangement of Eq. (1) allows to solve for $\frac{dC}{dt}$ as a function of $\frac{Q}{V}$; where $\frac{Q}{V}$ represents the inverse of the residence time, referred to here as λ, or the turnover rate

(i.e., $\lambda = \frac{Q}{V}$). Subsequently, Eq. (1) can also be rearranged to solve for the concentration in the control volume (C) as a function of time (t):

$$\frac{dC}{\lambda C_{eq} - \lambda C + \frac{G}{V}} = dt \tag{2}$$

Eq. (3) shows the integration of Eq. (2) over time:

$$\int_{C_0}^{C} \frac{dC}{\lambda C_{eq} - \lambda C + \frac{G}{V}} = \int_{t_0}^{t} dt \tag{3}$$

This results in:

$$C = C_{eq} + \left(C_0 - C_{eq}\right) F + \frac{G}{\lambda V}(1 - \mathrm{F}) \tag{4}$$

In the Eq. (4), F is calculated as follows:

$$F = \exp\left[-\lambda\left(\mathrm{t} - t_0\right)\right] \tag{5}$$

where t_0 and C_0 are the initial start time and concentration, dependent on the integration interval. Eq. (4) can be used to either estimate the concentration in the control volume with a known turnover rate or to calculate the average turnover rate using a simple linear regression with a known generation rate and other conditions.

EO is designed in this sub-section using the above equations as the overall framework. In EO, a particle is analogous to a solution and a concentration is analogous to a particle's position in the PSO algorithm. As Eq. (4) shows, there are three terms presenting the updating rules for a particle, and each particle updates its concentration via three separate terms. The first term is the equilibrium concentration, defined as one of the best-so-far solutions randomly selected from a pool, called the equilibrium pool. The second term is associated with a concentration difference between a particle and the equilibrium state, which acts as a direct search mechanism. This term encourages particles to globally search the domain, acting as explorers. The third term is associated with the generation rate, which mostly plays the role of an exploiter, or solution refiner, particularly with small steps, although it sometimes contributes as an explorer as well. Each term and the way they affect the search pattern is defined in the following.

3.1.1. Initialization and function evaluation

Similar to most meta-heuristic algorithms, EO uses the initial population to start the optimization process. The initial concentrations are constructed based on the number of particles and dimensions with uniform random initialization in the search space as follows:

$$C_i^{initial} = C_{min} + rand_i\left(C_{max} - C_{min}\right) \qquad i = 1, 2, \ldots..n \tag{6}$$

$C_i^{initial}$ is the initial concentration vector of the ith particle, C_{min} and C_{max} denote the minimum and maximum values for the dimensions, Rand $_i$ is a random vector in the interval of [0, 1], and n is the number of particles as the population. Particles are evaluated for their fitness function and then are sorted to determine the equilibrium candidates.

3.1.2. Equilibrium pool and candidates (C_{eq})

The equilibrium state is the final convergence state of the algorithm, which is desired to be the global optimum. At the beginning of the optimization process, there is no knowledge about the equilibrium state and only equilibrium candidates are determined to provide a search pattern for the particles. Based on different experiments under different type of case problems, these candidates are the four best-so-far particles identified during the whole optimization process plus another particle, whose concentration is the arithmetic mean of the mentioned four particles. These four candidates help EO to have a better exploration capability, while the average helps in exploitation. The number of candidates is arbitrary and based on type of the optimization problem. One might use other numbers of candidates (e.g. 3 or 5), which is consistent with the literature [6]. For example, GWO uses three best-so-far candidates (alpha, beta, and gamma wolves) to update the positions of the other wolves. However, using less than four candidates degrades the performance of the method in multimodal and composition functions but will improve the results in unimodal functions. More than four candidates will have the opposite effect. These five particles are nominated as equilibrium candidates and are used to construct a vector called the equilibrium pool:

$$\vec{C}_{eq,pool} = \left\{\vec{C}_{eq(1)}, \vec{C}_{eq(2)}, \vec{C}_{eq(3)}, \vec{C}_{eq(4)}, \vec{C}_{eq(ave)}\right\} \tag{7}$$

Each particle in each iteration updates its concentration with random selection among candidates chosen with the same probability. For instance, in the first iteration, the first particle updates all of its concentrations based on $\vec{C}_{eq(1)}$; then, in the second iteration, it may update its concentrations based on $\vec{C}_{eq(ave)}$. Until the end of the optimization process, each particle will experience the updating process with all of the candidate solutions receive approximately the same number of updates for each particle.

3.1.3. Exponential term (F)

The next term contributing to the main concentration updating rule is the exponential term (F). An accurate definition of this term will assist EO in having a reasonable balance between exploration and exploitation. Since the turnover rate can vary with time in a real control volume, λ is assumed to be a random vector in the interval of [0, 1].

$$\vec{F} = e^{-\vec{\lambda}(t - t_0)} \tag{8}$$

where time, t, is defined as a function of iteration ($Iter$) and thus decreases with the number of iterations:

$$t = \left(1 - \frac{Iter}{Max\_iter}\right)^{\left(a_2 \frac{Iter}{Max\_iter}\right)} \tag{9}$$

where $Iter$ and Max_iter present the current and the maximum number of iterations, respectively, and a_2 is a constant value used to manage exploitation ability. In order to guarantee convergence by slowing down the search speed along with improving the exploration and exploitation ability of the algorithm, this study also considers:

$$\vec{t}_0 = \frac{1}{\vec{\lambda}} \ln\left(-a_1 sign\left(\vec{r} - 0.5\right)\left[1 - e^{-\vec{\lambda}t}\right]\right) + t \tag{10}$$

where a_1 is a constant value that controls exploration ability. The higher the a_1, the better the exploration ability and consequently the lower exploitation performance. Similarly, the higher the a_2, the better the exploitation ability and the lower the exploration ability. The third component, $sign\,(r - 0.5)$, effects on the direction of exploration and exploitation. r is a random vector between 0 and 1. For all of the problems subsequently solved in this paper, a_1 and a_2 are equal to 2 and 1, respectively. These constants are selected through empirical testing of a subset of test functions. However, these parameters can be tuned for other problems as needed.

Eq. (11) shows the revised version of Eq. (8) with the substitution of Eq. (10) into Eq. (8):

$$\vec{F} = a_1 sign\left(\vec{r} - 0.5\right)\left[e^{-\vec{\lambda}t} - 1\right] \tag{11}$$

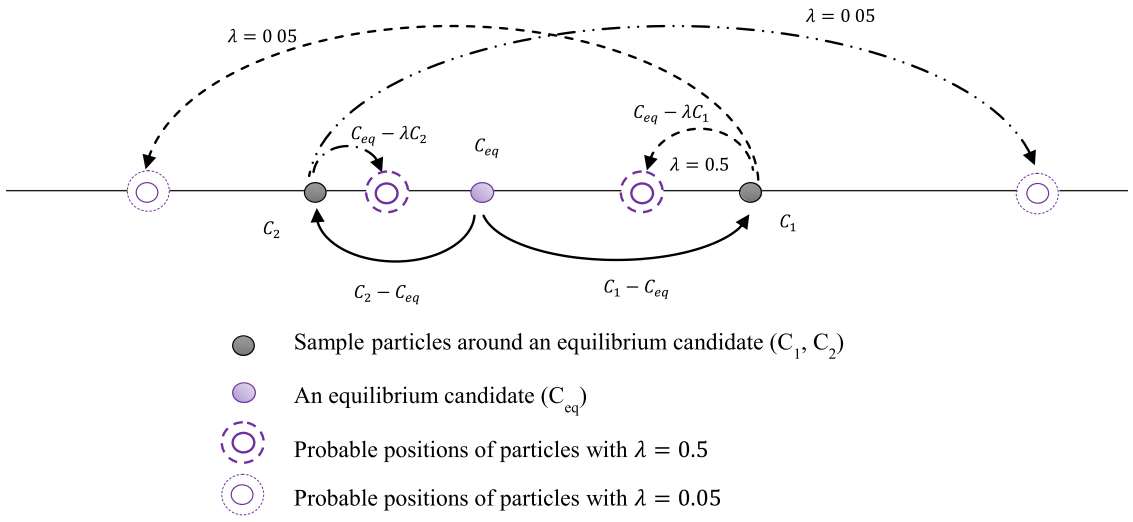

Fig. 1. 1-D presentation of concentrations updating aid in exploration and exploitation.

3.1.4. Generation rate (G)

The generation rate is one of the most important terms in the proposed algorithm to provide the exact solution by improving the exploitation phase. In many engineering applications, there are many models that can be used to express the generation rate as a function of time [22]. For example, one multipurpose model that describes generation rates as a first order exponential decay process is defined as:

$$\vec{G} = \vec{G}_0 e^{-\vec{k}(t-t_0)} \tag{12}$$

where G_0 is the initial value and k indicates a decay constant. In order to have a more controlled and systematic search pattern and to limit the number of random variables, this study assumes $k = \lambda$ and uses the previously derived exponential term. Thus, the final set of generation rate equations are as follows:

$$\vec{G} = \vec{G}_0 e^{-\vec{\lambda}(t-t_0)} = \vec{G}_0 \vec{F} \tag{13}$$

where:

$$\overrightarrow{G_0} = \overrightarrow{GCP}\left(\overrightarrow{C_{eq}} - \vec{\lambda}\vec{C}\right) \tag{14}$$

$$\overrightarrow{GCP} = \begin{cases} 0.5r_1 & r_2 \geq GP \\ 0 & r_2 < GP \end{cases} \tag{15}$$

where r_1 and r_2 are random numbers in [0, 1] and GCP vector is constructed by the repetition of the same value resulted from Eq. (15). In this equation, *GCP* is defined as the Generation rate Control Parameter, which includes the possibility of generation term's contribution to the updating process. The probability of this contribution which specifies how many particles use generation term to update their states is determined by another term called Generation Probability (GP). The mechanism of this contribution is determined by Eqs. (14) and (15). Eq. (15) occurs at the level of each particle. For example, if GCP is zero, G is equal to zero and all the dimensions of that specific particle are updated without a generation rate term. A good balance between exploration and exploitation is achieved with $GP = 0.5$. Finally, the updating rule of EO will be as follows:

$$\vec{C} = \vec{C}_{eq} + \left(\vec{C} - \vec{C}_{eq}\right).\vec{F} + \frac{\vec{G}}{\vec{\lambda}V}\left(1 - \vec{F}\right) \tag{16}$$

where F is defined in Eq. (11), and V is considered as unit.

The first term in Eq. (16) is an equilibrium concentration, where the second and third terms represent the variations in concentration. The second term is responsible for globally searching the space to find an optimum point. This term contributes more to exploration, thereby taking advantage of large variations in concentration (i.e., a direct difference between an equilibrium and a sample particle). As it finds a point, the third term contributes to making the solution more accurate. This term thus contributes more to exploitation and benefits from small variations in concentration, which are governed by the generation rate term (Eq. (13)). Depending on parameters such as the concentrations of particles and equilibrium candidates, as well as the turnover rate (λ), the second and third terms might have the same or opposite signs. The same sign makes the variation large, which helps to better search the full domain, and the opposite sign makes the variation small, aiding in local searches.

Although the second term attempts to find solutions relatively far from equilibrium candidates and the third term attempts to refine the solutions closer to the candidates, this is not always happening. Small turnover rates (e.g., ≤0.05) in the denominator of the third term increase its variation and helps the exploration in some dimensions as well. Fig. 1 demonstrates a 1-D version of how these terms contribute to exploration and exploitation. $C_1 - C_{eq}$ is representative of the second term in Eq. (16) while $C_{eq} - \lambda C_1$ represents the third term (G is the function of G_0). The generation rate terms (Eqs. (13)–(15)) control these variations. Because λ changes with each dimension's change, this large variation only happens to those dimensions with small values of λ. It is worth mentioning that this feature works similar to a mutation operator in evolutionary algorithms and greatly helps EO to exploit the solutions.

Fig. 2 shows a conceptual sketch of the collaboration of all equilibrium candidates on a sample particle and how they affect concentration updating, one after another, in the proposed algorithm. Since the topological positions of equilibrium candidates are diverse in initial iterations, and the exponential term generates large random numbers, this step by step updating process helps the particles to cover the entire domain in their search. An opposite scenario happens in the last iterations, when the candidates surround the optimum point by similar configurations. At these times, the exponential term generates small random numbers, which helps in refining the solutions by providing smaller step sizes. This concept can also be extended to higher dimensions as a hyperspace whereby the concentration will be updated with the particle's movement in n-dimensional space.

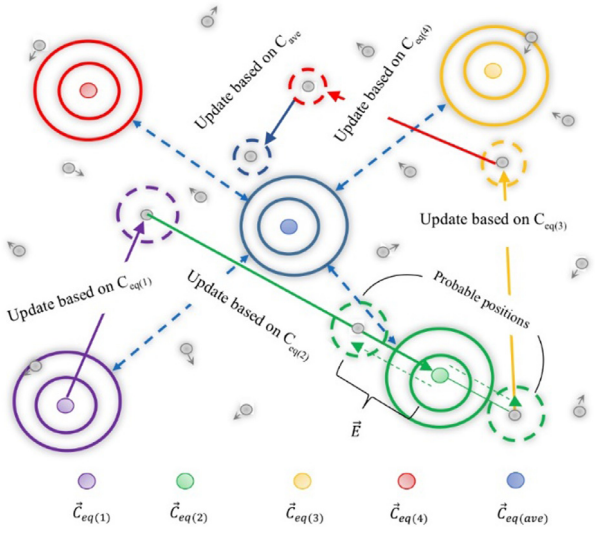

Fig. 2. Equilibrium candidates' collaboration in updating a particles' concentration in 2D dimensions.

3.1.5. Particle's memory saving

Adding memory saving procedures assists each particle in keeping track of its coordinates in the space, which also informs its fitness value. This mechanism resembles the *pbest* concept in PSO. The fitness value of each particle in the current iteration is compared to that of the previous iteration and will be overwritten if it achieves a better fit. This mechanism aids in exploitation capability but can increase the chance of getting trapped in local minima if the method does not benefit from global exploration ability [23]. The pseudo code of the proposed EO algorithm along with a memory saving function is presented in Fig. 3.

3.1.6. Exploration ability of EO

To summarize these terms, there are several parameters and mechanisms in EO that lead to exploration, as follows:

- a_1: controls the exploration quantity (magnitude) of the algorithm. It determines how far the new position would be to the equilibrium candidate. The higher the a_1 value, the higher the exploration ability. Note that numbers greater than three would considerably degrade the exploration performance. Since a_1 can magnify the concentration variation, it should be large enough to expand the exploration ability. However, based on empirical testing, it was found that values greater than three push the agents to search on boundaries. This recommendation is similar to the recommendation for free parameters in other algorithms. For example, in PSO, it is recommended that the sum of social and cognitive parameter should be less than or equal to four [24].
- $sign\,(r-0.5)$: controls the exploration direction. Since r is in [0, 1] with uniform distribution, there is equal probability of negative and positive signs.
- *Generation probability (GP):* controls the participation probability of concentration updating by the generation rate. $GP = 1$ means that there will be no generation rate term participating in the optimization process. This state emphasizes high exploration capability, and often leads to non-accurate solutions. $GP = 0$ means that the generation rate term will always be participating in the process, which increases the stagnation probability in local optima. Based on empirical testing, $GP = 0.5$ provides a good balance between exploration and exploitation phases.
- *Equilibrium pool*: This vector consists of five particles. The selection of five particles is somewhat arbitrary but was chosen based on empirical testing. In the initial iterations, the candidates are all far away from each other in distance. Updating the concentrations based on these candidates improves the algorithm's ability to globally search the space. The average particle also helps to discover unknown search spaces at initial iterations when particles are far away from each other.

3.1.7. Exploitation ability of EO

The main parameters and mechanisms to perform exploitation and local search in EO are as follows:

- a_2: this parameter is similar to a_1, but controls the exploitation feature. It determines the quantity (magnitude) of exploitation by digging around the best solution.
- $sign\,(r-0.5)$: controls the exploitation quality (direction) as well. It specifies the direction of a local search.
- *Memory saving*: memory saving, saves a number of best-so-far particles and substitutes them for worse particles. This feature directly improves the EO's ability for exploitation.
- *Equilibrium pool*: by lapse of iteration, exploration fades out and exploitation fades in. Thus, in the last iterations, where the equilibrium candidates are close to each other, the concentration updating process will aid in local search around the candidates, leading to exploitation.

3.1.8. Computational complexity analysis

Computational complexity of an optimization algorithm is presented by a function relating the running time of the algorithm to the input size of problem. For this purpose, Big-O notation is used here as a common terminology. Complexity is dependent upon the number of particles (n), the number of dimensions (d), and the number of iterations (t), and (c) is the cost of function evaluation.

$$\begin{aligned} O\,(EO) &= O\,(problem\ definition) + O\,(initialization) \\ &+ O(t\,(function\ evaluations)) + \\ O(t\,(Memory\ saving)) &+ O(t\,(Concentration\ Update)) \end{aligned} \tag{17}$$

Therefore, the overall computational complexity is defined as:

$$O\,(EO) = O\,(1 + nd + tcn + tn + tnd) \cong O\,(tnd + tcn) \tag{18}$$

As it is shown, the complexity is of the polynomial order. Thus, EO can be considered as an efficient algorithm. The complexity of EO with that of PSO and GA (as two of the most well-known meta-heuristics) is compared in Appendix A.

4. Results on benchmark functions

This section demonstrates the effectiveness of the proposed algorithm on a set of 58 benchmark test functions, including 29 commonly used unimodal, multimodal, and composition functions, as well as another 29 functions from the CEC-BC-2017 test suite [25]. This study utilizes both quantitative and qualitative validation metrics. Quantitative metrics include the average and standard deviation values for different test functions and qualitative metrics include trajectory, search, optimization, and average fitness history.

Initialize the particle's populations $i=1,\ldots,n$
Assign equilibrium candidates' fitness a large number
Assign free parameters $a_1=2$; $a_2=1$; $GP=0.5$;
While Iter < Max_iter
For i=1: number of particles (n)
Calculate fitness of i^{th} *particle*
If $fit(\vec{C}_i) < fit(\vec{C}_{eq\,1})$
Replace $\vec{C}_{eq\,1}$ *with* $\vec{C}_i$ *and fit* $(\vec{C}_{eq\,1})$ *with* $fit(\vec{C}_i)$
Elseif $fit(\vec{C}_i) > fit(\vec{C}_{eq\,1})$ & $fit(\vec{C}_i) < fit(\vec{C}_{eq\,2})$
Replace $\vec{C}_{eq\,2}$ *with* $\vec{C}_i$ *and fit* $(\vec{C}_{eq\,2})$ *with* $fit(\vec{C}_i)$
Elseif $fit(\vec{C}_i) > fit(\vec{C}_{eq\,1})$ & $fit(\vec{C}_i) > fit(\vec{C}_{eq\,2})$ & $fit(\vec{C}_i) < fit(\vec{C}_{eq\,3})$
Replace $\vec{C}_{eq\,3}$ *with* $\vec{C}_i$ *and fit* $(\vec{C}_{eq\,3})$ *with* $fit(\vec{C}_i)$
Elseif $fit(\vec{C}_i) > fit(\vec{C}_{eq\,1})$ & $fit(\vec{C}_i) > fit(\vec{C}_{eq\,2})$ & $fit(\vec{C}_i) > fit(\vec{C}_{eq\,3})$ & $fit(\vec{C}_i) < fit(\vec{C}_{eq\,4})$
Replace $\vec{C}_{eq\,4}$ *with* $\vec{C}_i$ *and fit* $(\vec{C}_{eq\,4})$ *with* $fit(\vec{C}_i)$
End (If)
End (For)
$\vec{C}_{ave} = (\vec{C}_{eq\,1} + \vec{C}_{eq\,2} + \vec{C}_{eq\,3} + \vec{C}_{eq\,4})/4$
Construct the equilibrium pool $\vec{C}_{eq,pool} = \{\ \vec{C}_{eq(1)}, \vec{C}_{eq(2)}, \vec{C}_{eq(3)}, \vec{C}_{eq(4)}, \vec{C}_{eq(ave)}\ \}$
Accomplish memory saving (if Iter > 1)
Assign $t = (1 - \frac{Iter}{Max\_iter})^{\left(a_2 \frac{Iter}{Max\_iter}\right)}$ *Eq (9)*
For i=1: number of particles (n)
Randomly choose one candidate from the equilibrium pool (vector)
Generate random vectors of $\vec{\lambda}$, $\vec{r}$ *from Eq (11)*
Construct $\vec{F} = a_1 sign(\vec{r} - 0.5)[e^{-\vec{\lambda}t} - 1]$ *Eq (11)*
Construct $\overrightarrow{GCP} = \begin{cases} 0.5r_1 & r_2 \geq GP \\ 0 & r_2 < GP \end{cases}$ *Eq (15)*
Construct $\overrightarrow{G_0} = \overrightarrow{GCP}(\overrightarrow{C_{eq}} - \vec{\lambda}\vec{C})$ *Eq (14)*
Construct $\vec{G} = \overrightarrow{G_0}.\vec{F}$ *Eq (13)*
Update concentrations $\vec{C} = \vec{C}_{eq} + (\vec{C} - \vec{C}_{eq}).\vec{F} + \frac{\vec{G}}{\vec{\lambda}V}(1 - \vec{F})$ *Eq (16)*
End (For)
Iter=Iter+1
End while

Fig. 3. Detailed pseudo code of EO.

4.1. Mathematical optimization test problems

Fig. 4 shows a two-dimensional version of the three categories of mathematical functions that this study uses to evaluate EO. The first category includes unimodal functions (F1–F7) that have a single optimum solution, which purposefully challenges the exploitation ability of the algorithm. The second category includes multi-modal functions (F8–F13) that have more than one optimal solution. Local optimal solutions in these functions evaluate the exploration performance of the algorithm, while an algorithm needs to be able to globally search the space and avoid being trapped in local optima in order to find the global optima. The third category includes fixed-dimensional multi-modal functions (F14–F23), which are similar to multi-modal functions but in low and fixed dimensions. These functions along with their dimensions employed in this study and constant coefficients are available in [6,17,25].

To further challenge the performance of EO, this study uses composite test functions (CF1–CF6) that mimic the complexity of a real search domain by having a large number of local optima and different shapes of the functions in different regions. Also, these functions are composed by shifting, rotating, expanding, and hybridizing unimodal and multimodal functions. Consequently, these functions represent more challenging optimization problems. More details about the composition function can be found in CEC 2005 technical report [26].

For all test categories, EO uses 30 particles along with 500 iterations (15,000 maximum function evaluations). Similarly, to provide a fair comparison, the other methods also use 15,000 maximum function evaluations. For example, GA used 30 populations along with 500 generations. The analogous terminology of this configuration for GWO is stated by 30 search agents associated with 500 iterations. In LSHADE-SPACMA, we used 18D (D is dimension) number of initial populations, but since the population linearly decreases over the iteration, we used 15,000 maximum function evaluations for a fair comparison. Table 1 shows the setting of parameters for each algorithm.

The comparative methods for this section include three categories of optimization methods: (i) Genetic Algorithm (GA) [7] and Particle Swarm Optimization (PSO) [8] as the most well-known and well-studied evolutionary and swarm intelligence algorithms; (ii) Gravitational Search Algorithm (GSA) [17], Grey Wolf Optimizer (GWO) [6], and Salp Swarm Algorithm (SSA) [15] as recent and effective meta-heuristics; and (iii) Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [13], Success-History Based Parameter Adaptation Differential Evolution (SHADE) (one of the CEC 2013 competitors) [27], and SHADE with

Table 1
Parameter settings for algorithms.

Algorithm	Parameter	Value
PSO	Topology	Fully connected
	Cognitive and social constant (C_1, C_2)	2, 2
	Inertia weight	Linear reduction from 0.9 to 0.1
	Velocity limit	10% of dimension range
GWO	Convergence parameter (a)	Linear reduction from 2 to 0
GA	Type	Real coded
	Selection	Roulette wheel (Proportionate)
	Crossover	Whole arithmetic (Probability = 0.8, $\alpha = [-0.5, 1.5]$)
	Mutation	Gaussian (Probability = 0.05)
GSA	Alpha, G_0, Rnorm, Rpower	20, 100, 2, 1
SSA	Leader position update probability	0.5
SHADE	Pbest, Arc rate	0.1, 2
LSHADE-SPACMA	Learning rate (c), threshold	0.8, max_nfes/2
	NP, H, Pbest, Arc rate,	18D, 5, 0.11, 1.4, (D = dimension size)
	Probability variable (Fcp)	0.5

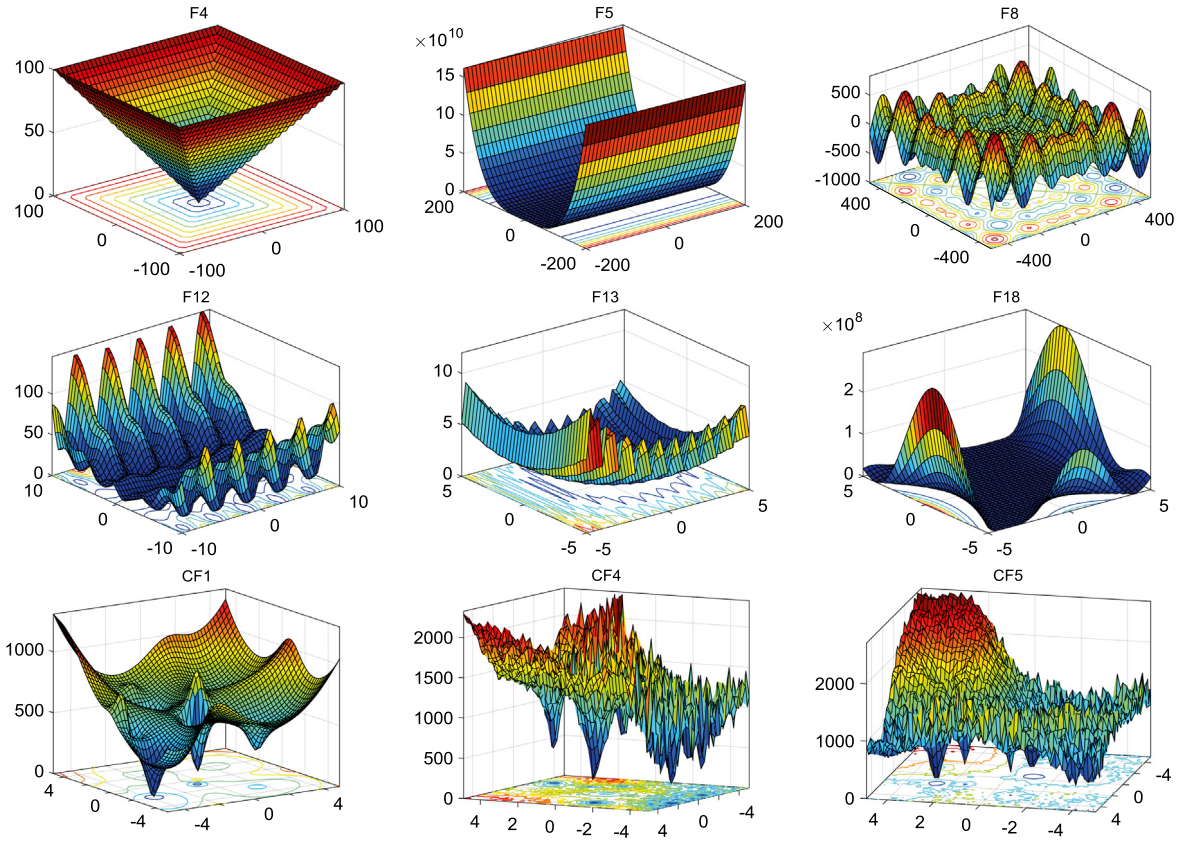

Fig. 4. A two-dimensional perspective view for couple of the mathematical benchmark functions.

linear population size reduction hybridized with semi-parameter adaptation of CMA-ES (LSHADE-SPACMA) (one of the CEC 2017 winners) as high performance optimizers. It is noted that the comparison for all algorithms is done with equal floating-point precision, so the difference between results are due to the performance of the method.

4.2. EO's performance on unimodal test functions

Unimodal functions are designed to test the exploitation ability of a method. From the results on unimodal functions in Table 2 (F1–F7), it is evident that EO outperformed almost all methods on the majority of functions. As it is seen, EO was able to achieve the first rank in unimodal test functions. This superior performance is also seen on both average and standard deviation in functions of F1, F2, F3, F4, and F7. In F5, EO received the second rank after SHADE with only a slight difference in the average outcome. For the other function (F6), the results of EO were competitive to the other methods. Based on the characteristics of unimodal functions, it can be stated that EO benefits from high exploitation ability.

Table 2
Optimization results and comparison for functions.

	Function		EO	PSO	GWO	GA	GSA	SSA	CMA-ES	SHADE	LSHADE-SPACMA
Unimodal	F1	Ave	3.32E−40	9.59E−06	6.59E−28	0.55492	2.53E−16	1.58E−07	1.42E−18	1.42E−09	0.2237
		Std	6.78E−40	3.35E−05	1.58E−28	1.23010	9.67E−17	1.71E−07	3.13E−18	3.09E−09	0.1480
	F2	Ave	7.12E−23	0.02560	7.18E−17	0.00566	0.05565	2.66293	2.98E−07	0.0087	21.1133
		Std	6.36E−23	0.04595	7.28E−17	0.01443	0.19404	1.66802	1.7889	0.0213	9.5781
	F3	Ave	8.06E−09	82.2687	3.29E−06	846.344	896.534	1709.94	1.59E−05	15.4352	88.7746
		Std	1.60E−08	97.2105	1.61E−05	161.499	318.955	11242.3	2.21E−05	9.9489	47.2300
	F4	Ave	5.39E−10	4.26128	5.61E−07	4.55538	7.35487	11.6741	2.01E−06	0.9796	2.1170
		Std	1.38E−09	0.67730	1.04E−06	0.59153	1.74145	4.1792	1.25e−06	0.7995	0.4928
	F5	Ave	25.32331	92.4310	26.81258	268.248	67.5430	296.125	36.7946	24.4743	28.8255
		Std	0.169578	74.4794	0.793246	337.693	62.2253	508.863	33.4614	11.2080	0.8242
	F6	Ave	8.29E−06	8.89E−06	0.816579	0.56250	2.5E−16	1.80E−07	6.83E−19	5.31E−10	0.2489
		Std	5.02E−06	9.91E−06	0.482126	1.71977	1.74E−16	3.00E−07	6.71E−19	6.35E−10	0.1131
	F7	Ave	0.001171	0.02724	0.002213	0.04293	0.08944	0.1757	0.0275	0.0235	0.0047
		Std	6.54E−04	0.00804	0.001996	0.00594	0.04339	0.0629	0.0079	0.0088	0.0019
Multimodal (High dimensional)	F8	Ave	−9016.34	−6075.85	−6123.1	−10546.1	−2821.1	−7455.8	−7007.1	−11713.1	−3154.4
		Std	595.1113	754.632	909.865	353.158	493.037	772.811	773.94	230.49	317.921
	F9	Ave	0	52.8322	0.31052	30.8229	25.9684	58.3708	25.338	8.5332	67.542
		Std	0	16.7068	0.35214	7.57295	7.47006	20.016	8.5539	2.1959	10.016
	F10	Ave	8.34E−14	0.00501	1.06E−13	1.63551	0.06208	2.6796	15.587	0.3957	0.0393
		Std	2.53E−14	0.01257	2.24E−13	0.46224	0.23628	0.8275	7.9273	0.5868	0.0151
	F11	Ave	0	0.02381	0.00448	0.56112	27.7015	0.0160	5.76E−15	0.0048	0.8948
		Std	0	0.02870	0.00665	0.26942	5.04034	0.0112	6.18E−15	0.0077	0.1078
	F12	Ave	7.97E−07	0.02764	0.05343	0.03088	1.79961	6.9915	2.87E−16	0.0346	8.18E−04
		Std	7.69E−07	0.05399	0.02073	0.04092	0.95114	4.4175	5.64E−16	0.0875	0.0010
	F13	Ave	0.029295	0.00732	0.65446	0.36222	8.89908	15.8757	3.66E−04	7.32E−04	0.0102
		Std	0.035271	0.01050	0.00447	0.30975	7.12624	16.1462	0.0020	0.0028	0.0103
Multimodal (Fixed-dimensional)	F14	Ave	0.998004	3.84902	4.042493	0.998004	5.859838	1.1965	10.237	0.998004	1.9416
		Std	1.54E−16	3.24864	4.252799	4.23E−12	3.831299	0.5467	7.5445	5.83E−17	2.9633
	F15	Ave	0.002398	0.002434	0.00337	0.005206	0.003673	0.000886	0.0057	0.002374	3.00E−04
		Std	0.006097	0.006081	0.00625	0.007028	0.001647	0.000257	0.0121	0.0061	1.93E−19
	F16	Ave	−1.03162	−1.03162	−1.03163	−1.03162	−1.03163	−1.03163	−1.03162	−1.03162	−1.03162
		Std	6.04E−16	6.51E−16	2.13E−08	1.34E−06	4.88E−16	6.13E−14	6.77E−16	6.51E−16	1.00E−15
	F17	Ave	0.397887	0.397887	0.397889	0.397890	0.397887	0.397887	0.397887	0.397887	0.397887
		Std	0	0	2.13E−04	1.08E−05	0	3.41E−14	0	3.24E−16	0
	F18	Ave	3	3	3.000028	3.000002	3	3	8.4000	3	3
		Std	1.56E−15	1.97E−15	4.24E−04	4.06E−06	4.17E−15	2.20E−13	20.550	1.87E−15	1.25E−15
	F19	Ave	−3.86278	−3.86278	−3.86263	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278
		Std	2.59E−15	2.65E−15	0.00273	1.63E−07	2.29E−15	1.47E−10	2.7E−15	2.69E−15	2.7E−15
	F20	Ave	−3.2687	−3.26651	−3.28654	−3.27443	−3.31778	−3.2304	−3.2903	−3.27047	−3.28234
		Std	0.05701	0.06032	0.10556	0.05924	0.023081	0.0616	0.0535	0.0599	0.0570
	F21	Ave	−8.55481	−5.9092	−8.7214	−5.72536	−5.95512	−9.6334	−5.6642	−9.2343	−9.4735
		Std	2.76377	3.59559	2.6914	3.32622	3.73707	1.8104	3.3543	2.4153	1.7626
	F22	Ave	−9.3353	−7.3360	−9.2415	−6.94349	−10.4015	−9.0295	−8.4434	−10.1479	−10.2258
		Std	2.43834	3.47381	1.61254	3.56118	2.01408	2.3911	3.3388	1.3969	0.9704
	F23	Ave	−9.63655	−8.7482	−10.5343	−7.0208	−10.5364	−9.0333	−8.0750	−10.2809	−10.5364
		Std	2.38811	2.55743	0.00125	3.85233	2.6E−15	2.9645	3.5964	1.3995	1.77E−15
Composition	F24 (CF1)	Ave	66.666	151.18	90.229	86.671	20.000	43.333	209.48	63.333	3.3333
		Std	95.893	123.49	105.51	97.324	48.423	67.891	215.06	80.872	18.254
	F25 (CF2)	Ave	89.837	204.92	163.56	142.72	186.77	31.133	189.83	40.508	0.0000
		Std	56.366	118.89	89.476	119.58	62.726	52.149	170.79	61.462	0.0000
	F26 (CF3)	Ave	161.73	273.73	210.61	214.67	218.55	235.11	274.20	139.48	104.29
		Std	33.227	110.87	95.214	73.470	117.02	80.839	213.89	33.366	14.266
	F27 (CF4)	Ave	356.44	487.45	418.63	447.01	492.33	232.44	372.99	316.62	278.63
		Std	115.66	151.15	156.16	112.34	99.549	43.643	152.12	96.752	7.0670
	F28 (CF5)	Ave	52.309	214.56	143.81	91.831	232.32	27.538	224.85	39.515	2.02E−17
		Std	95.565	180.03	149.12	73.898	75.405	41.598	286.23	51.233	7.69E−17
	F29 (CF6)	Ave	768.48	794.50	837.47	811.21	845.47	628.69	845.26	684.51	540.23
		Std	192.94	175.94	136.45	173.11	80.524	184.48	139.52	201.22	122.75
Friedman mean rank			2.86	5.95	5.14	6.59	5.62	6.00	5.47	3.47	3.91
Rank			1	7	4	9	6	8	5	2	3

4.3. EO's performance on multimodal test functions

Multimodal functions can evaluate the exploration ability of an algorithm because of their high number of local optima. The number of locally optimal solutions in these types of functions exponentially increases in proportion to the number of dimensions. The results of EO on multimodal functions are given in Table 2 for high dimensional (F8–F13) and for fixed dimensional functions (F14–F23). The table shows that EO outperformed other methods on F9, F10, and F11 high dimensional problems. It is worth noting that on F9 and F11, EO was able to obtain the global optimum while most other methods did not. For F8, which is the most difficult function in this class, EO obtained the second rank after SHADE, which reached almost to the global optimum. For F12 and F13, CMA-ES showed better performance than other methods, while still EO showed competitive results especially for F12. It is important to note that in these functions, EO achieved the second-best performance after CMA-ES and actually ranked first in this class overall. In fixed dimensional problems, the performance of all methods was similar, and the results for EO are very competitive with the others. In functions F14, F16, F17, F18, and F19 in this class, EO reached the global optimum. For the remaining functions, EO's results were very close to the global optimum.

4.4. EO's performance on composition functions

Composite test cases are the most challenging test beds. These functions are designed to evaluate local minima avoidance performance as well as the exploration ability of a method. Table 2 (F24–F29) shows the performance of EO on composition functions compared to other methods. Overall, LSHADE-SPACMA showed better performance compared to other methods and ranked first in this category. In most functions, EO ranked fourth among these methods after LSHADE-SPACMA, SHADE, and SSA which are among the high-performance and recent optimizers. Although EO gained the fourth rank in this class, its results are competitive to SHADE and SSA in most functions. Since this class of functions are associated with too many local minima, in order to have a better understanding of distribution of results, the boxplot of results for each algorithm and function are shown in Fig. 5. As an example, the local minima stagnation in all algorithms is evident in CF6 of Fig. 5. Based on its boxplot, the results are clustered in two groups around 500 and 900. This is due to the complex topology of this function which makes the algorithms get stuck in local minima. This behavior is more or less observable in CF4 as well.

To summarize, based on the results of Table 2, EO showed excellent performance, on unimodal and multimodal functions and very good performance in fixed-dimensional multimodal functions among the other methods compared. The performance of EO in the composition class of functions was lower compared to previous classes. However, it is important to note that SHADE and LSHADE-SPACMA utilize a complex strategy and they are among the strongest competitors and winners of CEC 2013 and 2017, respectively. Based on the Friedman mean rank and overall rank, EO placed as the best performing method (ranked first) among methods considering all types of functions. The mean rank shows that the performance of SHADE and LSHADE-SPACMA is close to EO while performance of these three algorithms is by far better than all other algorithms. EO's overall satisfying performance in this challenging class of functions is attributable to the intermittent contribution of the generation rate term for concentration updating. Therefore, the results of this testing indicate that EO is a powerful and robust optimizer designed to conduct well-tuned exploration and exploitation.

4.5. Sensitivity analysis of EO's parameters

This section analyzes the sensitivity of three control parameters of EO, which are a_1, a_2, and GP. This analysis indicates which parameters are robust and which parameters are sensitive to different inputs, and which parameters affect the performance of the algorithm. During the development of EO, this study performed a full factorial design with these parameters on one function from each category of unimodal, multimodal, and fixed-dimensional multimodal functions. The first function was picked from each category (i.e., F1, F8, and F14). The values of each parameter for the factorial design are defined as $a_1 = \{1, 1.5, 2, 2.5, 3\}$, $a_2 = \{0.1, 0.5, 1, 1.5, 2\}$, $GP = \{0.1, 0.25, 0.5, 0.75, 0.9\}$. Since each of the 3 parameters have 5 values, the full factorial will have 5*5*5 = 125 combinations of design. Each design is the fitness average of functions with maximum of 15,000 function evaluations and 30 independent runs. Fig. 6 shows this sensitivity analysis for the 3 different functions. The x-axis shows the three control parameters and their associated values while the y-axis shows the average fitness for the functions. In F1 (Fig. 6a), a_2 shows sensitivity to its left boundary while a_1 and GP have the similar sensitivity to their right boundary, but all parameters showing robust behavior for their three middle values. In F8 (Fig. 6b), a_1 shows a sensitive behavior to all values and having the best performance on its middle value ($a_1 = 2$) while other parameters of a_2 and GP acted robust on their different range. As showed in F1, the sensitivity is the lowest in their middle range. In F14 (Fig. 6c), a_1 showed a high sensitivity in its left boundary while showing steady behavior for its right boundary. a_2 and GP showed robust behavior for different values. The overall behavior indicates that a_1 is more sensitive than a_2 and GP. Based on this analysis, the best value for a_1 can be suggested as its middle value (i.e., $a_1 = 2$). For a_2 and GP, which are more robust than a_1, any values other than their left and right boundaries can be selected. Being more conservative, the middle values are selected for these parameters (i.e., $a_2 = 1$ and $GP = 0.5$) since the neighboring (left and right) boundaries to the middle value shows less sensitivity compared to other selections. This experiment also suggests that the design of EO accounted for the best value of EO's control parameters in previous sections.

4.6. Convergence analysis of EO

Fig. 7 illustrates the five qualitative metrics used to evaluate efficiency and effectiveness of EO when solving real problems: trajectory, search, optimization, and average fitness history. These metrics are evaluated by solving mathematical functions with an intentionally reduced number of particles/iterations (10/100) to show a clear pattern of search and how particles in EO contribute to finding optimum points. The dimension of the problems did not change and remained intact. Although these functions have high dimensions, the presented 2D views of the functions in Fig. 7 provide insights on the domain's topology. The first qualitative metric discussed here is the search history, which includes the concentration (position) of particles from the first iteration to the last iteration. The concentration is shown on the counter lines of the search space for better understanding of how particles are able to explore and exploit the domain. Search history can also reveal the pattern that the particles used to search the space. The search history results show that the particles tend to aggregate around the optimum point in unimodal functions more effectively than in multimodal and composite functions. This behavior of EO demonstrates its exploitation ability, while the appropriate scattering of particles in the search space of multimodal and composition functions shows the promising capability of EO's exploration methods.

The second metric employed here is the optimization history (convergence curve). This metric is the fitness of the best-so-far

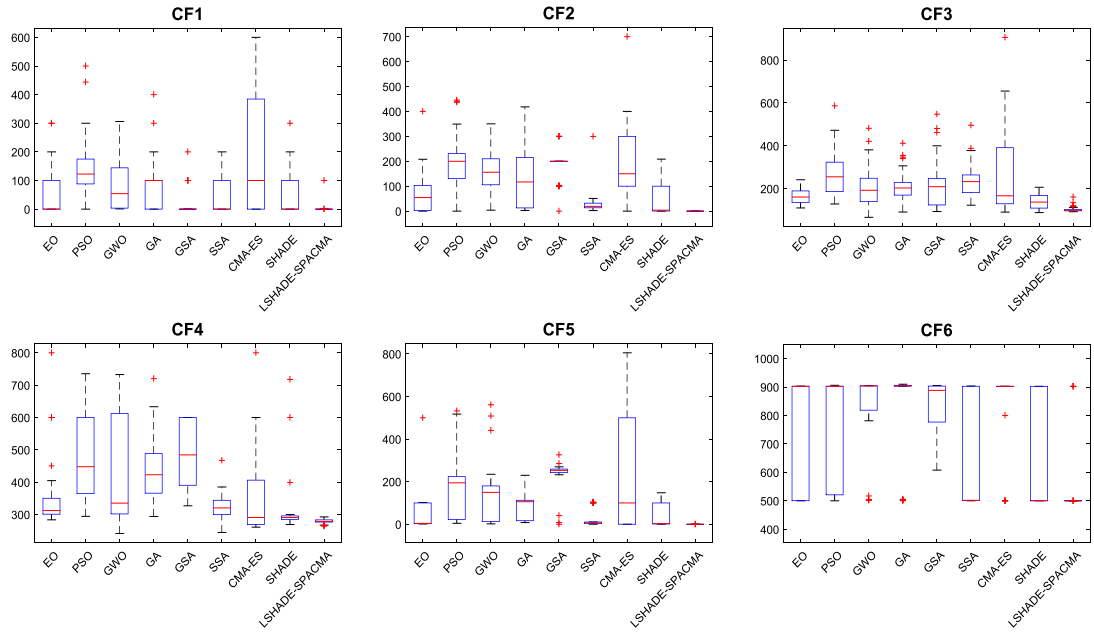

Fig. 5. Boxplot of composition functions results for different algorithms.

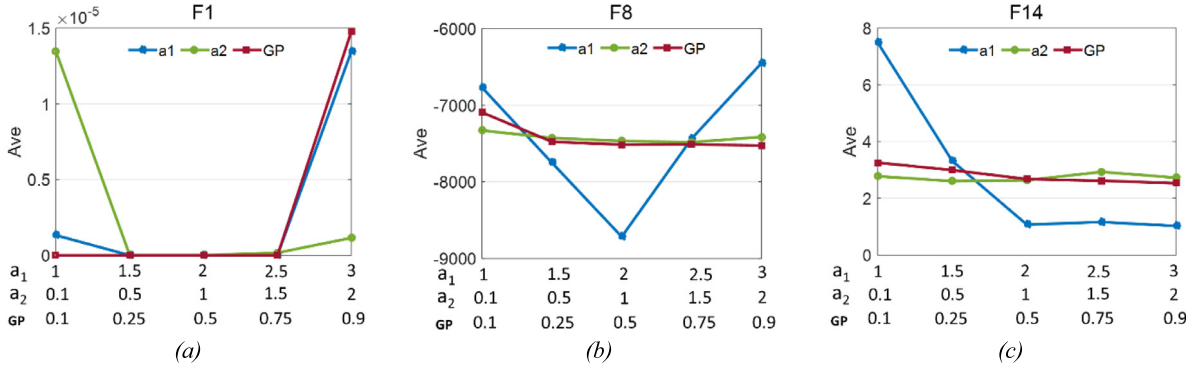

Fig. 6. Sensitivity analysis of EO's control parameters with different functions for (a) F1, (b) F8, and (c) F14.

particle ($C_{eq(1)}$) from the first to the last iteration. It shows how the global optimum is approximated by the algorithm with the lapse of iteration. There are various behaviors of optimization history for different types of functions. For unimodal functions, the curve is relatively smooth; for multimodal functions with high numbers of local optima, the behavior changes from smooth to step-like manner. This means that the algorithm has no improvement on that specified course of iterations due to the complexity of the domain, which can be seen in F8, F14, and CF2.

Although the history shows how the global optimum is estimated by exploration and exploitation, it does not provide insight on the total behavior of particles participating in the optimization process to improve the result. This trend is shown by the average fitness history in the fourth column of Fig. 7. The descending trend of all history curves shows that all the particles are collaborating to improve the results by updating their concentrations to better ones as the iteration number increases. The stability of the curves is due to the memory saving process, which does not allow particles to move toward the regions with lower fitness. Omitting the memory saving feature will lead to fluctuating behavior of average fitness curves. For all of the test functions there is a close similarity between the average fitness and optimization history curves. This similarity reveals a search pattern that all the particles are following, a pattern which emphasizes collaborative searching rules rather than independent acting, led by equilibrium candidates.

Another qualitative metric is the trajectory of particles, shown in column 5 of Fig. 7, which shows the variation of concentration (position) in the first dimension of the first particle. In all of the test functions, there is a sudden and abrupt oscillation in the first iterations, followed by a fade out in the last iterations. These sudden changes demonstrate exploration behavior by globally searching the domain at the initial iterations, while subsequently stabilizing by local searching as the iteration number increases. According to Berg et al. [28], this trend can guarantee that an algorithm will finally converge to a global/local optimum point. Following the trajectory curves, one can notice that the fluctuations are related to the domain complexity. The more complex the domain, the more fluctuations occur. In unimodal functions, there are no fluctuations after a number of iterations. In multimodal functions, fluctuations are greater in magnitude and frequency and persist for greater numbers of iterations. This behavior demonstrates that EO benefits from well-tuned exploitation and exploration.

Although there are some direct and explicit methods such as ancestry tree-based approach [29] for measuring exploration and exploitation, most researcher use indirect approaches such

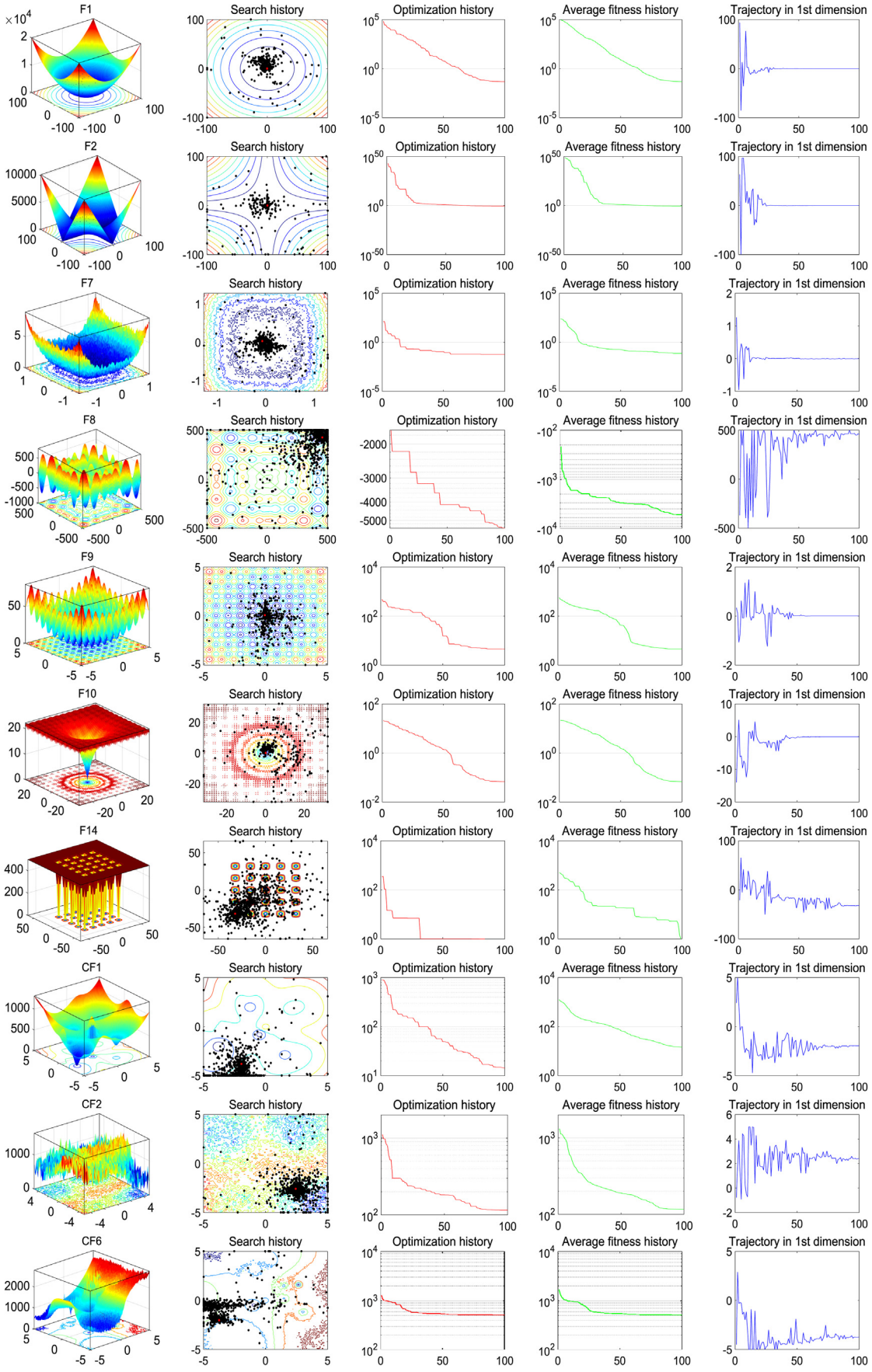

Fig. 7. Qualitative metrics: search history, optimization history, average fitness history and trajectory in 1st dimension.

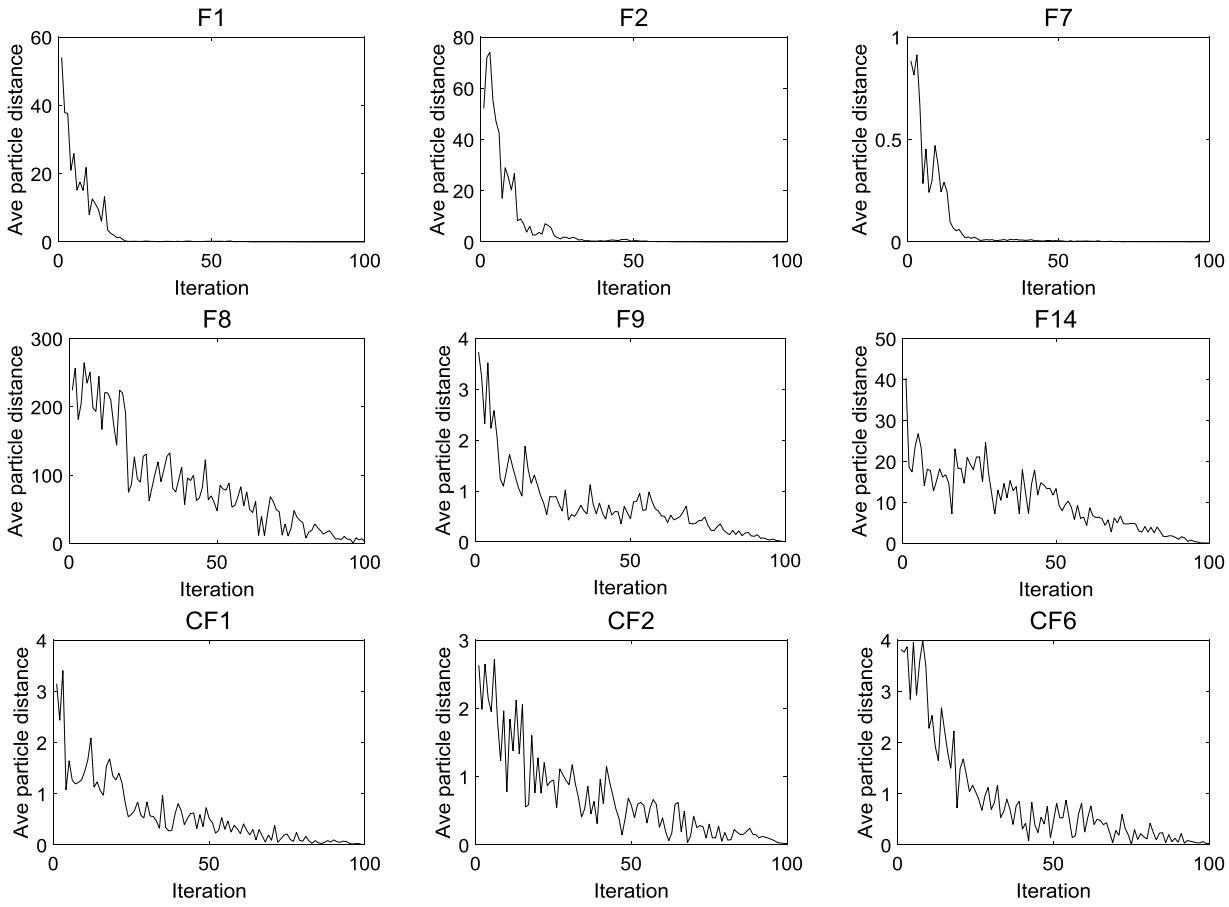

Fig. 8. Diversity history for unimodal, multimodal and composition functions.

as diversity to measure these features [30]. From exploration–exploitation viewpoint, an increase in diversity means that solutions are very different, which indicates that an algorithm is in exploration phase, while a decrease means solutions are within a certain neighborhood, thus showing exploitation behavior [30]. In order to depict a balanced exploration and exploitation in EO, the diversity histories are presented as the final qualitative metric in Fig. 7. The horizontal axis shows the number of iterations and the vertical axis indicates the average distance between particles (defined as the difference between concentrations). There is a decreasing distance trend in all curves in Fig. 8, again demonstrating a transient shift from exploration in initial iterations to exploitation in the final iterations. In unimodal functions (F1, F2, F3), this shift occurred very rapidly since EO is able to recognize the global optimum and tries to exploit it further. In multimodal and composite functions, EO showed an oscillating behavior but still kept the overall trend. This behavior is due to a large number of extrema in these functions. This metric can be used as further evidence for a well-defined balance between exploration-exploitation of the proposed EO algorithm.

4.7. Comparative convergence analysis

The previous section analyzed the convergence behavior of EO using defined qualitative metrics. This section compares the convergence behavior of other methods along with the EO algorithm. In order to have a concise and efficient comparison between algorithms, this study picked the best performing algorithm in each method category, including PSO in the category of most well-studied algorithms, GWO in the recent methods category, and SHADE in the high-performance optimizers category. This selection is based on the lowest obtained mean ranked among all 29 functions. For the functions, the first function of each category was used: F1 from unimodal, F8 from multimodal, F1 from fix dimensional, and CF1 from composition functions. Figs. 9 and 10 show qualitative metrics for the mentioned methods and functions. Since trajectory and diversity have fluctuating behaviors, they are depicted in a separate figure for each algorithm for easy following of the readers.

In F1, which is a unimodal function, EO and GWO were more successful than PSO and SHADE in recognizing the optimum point and reached better results with fewer iterations. The trajectory and diversity of EO and GWO are similar to each other and have more stable behavior than PSO and SHADE. In the multimodal function (F8), although SHADE has better results than other methods, as shown in the optimization history, the average fitness history of EO has better performance than other methods. It is due to the memory saving feature of EO, which does not allow particles to move toward less fitted regions. The trajectory and diversity of EO and SHADE in this function has more aggressive behavior than PSO and GWO, meaning that they were more successful in exploring the domain. Based on the optimization history of F14 depicted in Fig. 10, EO and SHADE estimated the optimum point in fewer iterations compared to PSO and GWO. One can notice a sudden decrease of fitness for algorithms in the last iterations shown in average fitness history. This decrease is due to the deep and narrow optimum points shaping the topology of F14, which is observable in Fig. 7. In the last iterations where algorithms try to search the neighborhood, gathering the search agents in these regions considerably improve the fitness of search agents.

Finally, for CF1, the optimization and average fitness history show that EO and SHADE performed better than PSO and GWO.

Table 3
Optimization results and comparison for CEC-BC-2017 test functions.

Function		EO	PSO	GWO	GA	GSA	SSA	CMA-ES	SHADE	LSHADE-SPACMA
CEC-2017-f1	Ave	2465.3	3959.6	325 132	9799.7	296.0	3396.25	100.00	100.00	100.00
	Std	2206.2	4456.6	107 351	5942.54	275.1	3673.08	0.000	0.000	0.000
CEC-2017-f3	Ave	300.00	300.00	1538.0	8721.4	10829.2	300.00	300.00	300.00	300.00
	Std	2.4E−08	1.9E−10	1886.02	5900.30	1620.74	0.00	0.000	0.000	0.000
CEC-2017-f4	Ave	404.48	405.94	409.5	410.71	406.6	406.27	400.00	400.00	400.00
	Std	0.7911	3.28	7.55	18.512	2.92	10.07	0.000	0.000	0.000
CEC-2017-f5	Ave	510.73	513.06	513.5	516.32	556.7	521.82	530.18	503.76	502.3
	Std	3.6707	6.54	6.10	6.926	8.40	10.50	58.32	1.006	0.87
CEC-2017-f6	Ave	600.00	600.24	600.6	600.04	621.6	609.77	682.1	600.00	600.00
	Std	1.5E−04	0.98	0.88	0.0668	9.015	8.26	35.43	2.6E−07	2.59E−07
CEC-2017-f7	Ave	720.93	718.98	729.8	728.32	714.6	740.88	713.4	713.90	711.32
	Std	5.7425	5.10	8.60	7.290	1.55	16.62	1.63	1.23	0.37
CEC-2017-f8	Ave	809.51	811.39	814.3	820.72	820.5	823.45	828.9	803.80	801.34
	Std	2.9176	5.47	8.26	8.961	4.69	9.95	52.98	1.27	1.03
CEC-2017-f9	Ave	900.00	900.00	911.3	910.28	900.0	944.07	4667.3	900.00	900.00
	Std	0.0227	5.9E−14	19.53	15.154	6.9E−14	104.66	2062.8	0	0
CEC-2017-f10	Ave	1418.7	1473.3	1530.5	1723.3	2694.6	1858.85	2588.1	1193.6	1047.2
	Std	261.63	214.97	286.67	252.34	297.62	294.50	414.47	84.7	56.55
CEC-2017-f11	Ave	1105.2	1110.5	1140.2	1125.6	1134.7	1180.5	1111.3	1100.8	1100.0
	Std	5.0218	6.28	54.13	23.80	10.45	59.80	25.44	1.36	4.2E−14
CEC-2017-f12	Ave	10 340	14 532	625 182	37 255	702 723	1 983 166	1629.6	1324.5	1341.7
	Std	9790.6	11 260	1 126 443	34792.7	42075.4	1 909 901	198.11	102.6	86.25
CEC-2017-f13	Ave	8023.0	8601.1	9842.3	10 828	11 053	16098.6	1323.6	1304.7	1303.7
	Std	6720.8	5123.6	5633.43	8928.94	2110.55	10537.2	78.32	0.71	3.25
CEC-2017-f14	Ave	1463.3	1482.1	3403.53	7048.9	7147.5	1508.94	1452.1	1410.8	1400.2
	Std	32.498	42.46	1953.33	8160.08	1489.52	51.05	55.98	9.21	0.44
CEC-2017-f15	Ave	1585.6	1714.3	3806.60	9296.2	18 001	2236.69	1509.6	1500.3	1500.3
	Std	48.012	282.89	3860.66	8978.18	5498.67	571.19	16.43	0.36	0.20
CEC-2017-f16	Ave	1649.0	1860.0	1725.78	1786.3	2149.7	1726.26	1815.3	1602.5	1600.9
	Std	50.915	127.65	123.85	129.07	105.8	126.97	230.1	2.19	0.36
CEC-2017-f17	Ave	1731.6	1761.6	1759.61	1746.5	1857.7	1774.57	1830.1	1716.4	1700.4
	Std	18.071	47.50	31.29	39.78	108.32	34.23	175.8	5.96	0.35
CEC-2017-f18	Ave	12 450	14 599	25806.1	15 721	8720.5	23429.1	1825.9	1809.9	1801.1
	Std	11 405	11852.2	15766.9	12 828	5060.1	14045.7	13.53	9.46	3.67
CEC-2017-f19	Ave	1951.5	2602.8	9866.1	9686.5	13 670	2916.1	1920.5	1900.5	1900.3
	Std	47.108	2185.02	6371.09	6766.3	19 168	1871.2	28.68	0.28	0.39
CEC-2017-f20	Ave	2020.6	2085.1	2075.6	2056.5	2272.3	2089.3	2494.8	2020.0	2000.2
	Std	22.283	62.25	52.04	60.01	81.72	49.28	242.65	0.0093	0.14
CEC-2017-f21	Ave	2307.5	2281.7	2317.1	2303.8	2357.7	2249.8	2324.7	2282.5	2202.1
	Std	20.961	54.02	7.00	43.75	28.20	60.44	67.76	42.6	4.02
CEC-2017-f22	Ave	2297.4	2314.8	2310.1	2304.6	2300.0	2301.5	3532.4	2297.3	2300.1
	Std	18.402	66.10	16.75	2.38	0.072	11.80	847.6	16.18	0.17
CEC-2017-f23	Ave	2615.8	2620.8	2616.4	2632.9	2736.5	2621.7	2728.8	2608.1	2603.3
	Std	5.5298	9.23	8.47	13.42	39.14	8.69	243.1	1.71	1.41
CEC-2017-f24	Ave	2743.8	2692.2	2741.7	2758.3	2742.2	2733.2	2704.4	2728.9	2677.9
	Std	6.904	108.20	8.73	14.92	5.52	64.43	73.42	31.7	91.6
CEC-2017-f25	Ave	2934.3	2924.0	2938.0	2947.9	2937.5	2923.5	2932.01	2916.4	2930.0
	Std	19.76	25.02	23.61	19.25	15.36	23.86	20.87	22.9	21.3
CEC-2017-f26	Ave	2967.8	2952.1	3222.5	3112.1	34407.5	2900.9	3457.7	2909.2	2900.0
	Std	164.98	249.66	427.02	334.65	628.73	36.56	598.9	34.9	0
CEC-2017-f27	Ave	3091.3	3116.2	3104.1	3115.1	3259.5	3092.6	3137.5	3071.5	3089.5
	Std	2.2414	24.99	21.81	19.18	41.66	2.78	21.37	0.78	0.15
CEC-2017-f28	Ave	3302.7	3315.9	3391.2	3320.7	3459.4	3210.5	3397.6	3266.7	3125.0
	Std	133.92	121.83	101.5	126.34	33.84	113.17	131.3	22.2	63.2
CEC-2017-f29	Ave	3169.9	3203.8	3190.5	3253.5	3449.5	3214.1	3213.5	3142.8	3134.9
	Std	24.65	52.26	42.9	81.99	171.33	51.69	109.79	12.9	3.87
CEC-2017-f30	Ave	297 113	350 650	297 688	537 277	1 303 361	421 120	304 569	3201.1	3430.6
	Std	458 560	504 857	527 757	637 410	363 843	568 085	444 815	0.31	33.45
Friedman mean rank		3.97	4.97	6.38	6.69	7.69	6.10	5.72	2.03	1.45
Rank		3	4	7	8	9	6	5	2	1

The high magnitude and frequency of oscillation in trajectories of EO and SHADE also demonstrate key differences. The diversity plot of PSO and GWO in this function fades out more rapidly than EO and SHADE. Overall, based on behavioral analysis of Figs. 8–10, SHADE has the best exploration ability among the compared methods. However, some diversity plots of SHADE show that it cannot show low diversity in last iterations compared to other

methods, which is why SHADE does not perform well in unimodal functions. Among the compared methods, PSO has the lowest exploration ability, showing fewer fluctuations in trajectory and damping sooner in diversity compared to the other methods. The analyses also indicate that EO showed more balanced behavior, having high fluctuations in initial iterations and low fluctuations in the last iterations in trajectory, and a well-balanced transient from high diversity to low diversity during the course of iterations. While the EO's exploration ability is not comparable to SHADE, the SHADE's exploitation ability is not as good as EO. However, EO's combined exploration and exploitation ability is better than GWO and PSO. The more detailed comparative behavioral analysis with all methods and functions can be considered as independent study.

4.8. Scalability analysis of EO

This section evaluates the performance of the EO algorithm for low- and high-dimensional problems using a scalability analysis. Since real-world optimization problems often involve a large number of variables, the algorithm starts from 10 to 200 dimensions with a step size of 10. To further challenge the performance and demonstrate the efficiency of the proposed algorithm, the test is carried out with a fixed number of iterations (500) and particles (30) as the dimension increases. This test shows how efficiently the algorithm will perform proportional to the dimension increase, while the number of iterations and particles both remain fixed.

The test is conducted on two samples of unimodal and multimodal functions to assess the exploration and exploitation ability. The results are compared with those of most well-known algorithms like GA, PSO and GWO as well. Fig. 11 shows the results for two unimodal (F1, F7) and multimodal (F8, F9) functions in logarithmic scale. Clearly, the performance of EO is reduced as the dimension increases. Since the iteration and particles are fixed, this trend is expected. In F1, all algorithms reach close to a steady state as the dimension increases. This shows that the performance of all algorithms is not substantially reduced when dealing with large numbers of variables, while EO has the best trend compared to the others. In another unimodal function, F7, EO performance is similar to GWO at low dimensions, while EO outperforms GWO at higher dimensions (and both are better than PSO and GA). In F8, a multimodal function, EO outperformed GWO and PSO, but had slightly worse performance than GA in low dimensions yet surpassed GA by dimension 50. Since this function has a negative global optimum and the magnitude of the optimum point increases linearly with dimension ($-418.98 \times d$), the steady behavior with increased dimensionality is not desired here. In other words, as the dimension increases, the expectation is to observe a decrease in fitness with linear behavior. Although all of the tested algorithms in this case have a similar overall behavior, EO again showed a better performance than the others. In the last function, F9, EO demonstrates a very interesting behavior. In all dimensions, EO was able to reach the global optimum point (zero) with perfect performance (and thus it is not shown in Fig. 11). The scalability test of EO for some samples of unimodal and multimodal functions boast its robustness in exploitation and exploration.

To summarize these comparisons, the convergence and scalability analysis along with the prior results comparison demonstrates EO as a potential candidate to face and solve mathematical optimization problems. The convergence analysis shows the capability of EO in finding optimum results from a completely random set of solutions, while the scalability analysis demonstrates its robustness with a reliable behavior in finding promising solutions toward the global optimum, even in very high dimensional problems.

4.9. EO's performance on CEC-BC-2017 test functions

In order to further challenge the EO algorithm, a more recent and challenging test suite, CEC-BC-2017, which includes shifted and rotated unimodal, multimodal, hybrid, and composition functions [31], was employed. It is also worth noting that due to unstable behavior, the f2 function has been removed from this test suite [32]. The search domain for all functions are [−100, 100] considering 10 dimensions. EO's performance was tested against this test suite and the results are compared with the same metaheuristics algorithms used previously. The majority of functions presented in this test suite are among more challenging hybrid and composition functions. The results for all algorithms are provided based on 50 search agents (populations) along with 1000 iterations associated with 30 independent runs. The parameter setting for each method was the same as mentioned in Table 1. Table 3 shows the optimization results of different methods in this test suite.

The overall rank that was achieved by EO in this test bed was third, following LSHADE-SPACMA and SHADE. This achievement is another demonstration of EO's ability to outperform well-studied and recent optimizers while also having very competitive results with high performance methods in standard functions.

4.10. Statistical analysis of EO

This section performed multiple statistical analysis tests to first understand if the differences in performance of all algorithms in the benchmark functions are statistically significant using the non-parametric Friedman test. In order to have a reliable comparison, there is a need to compare more than 10 benchmark functions with more than 5 different algorithms [33]. In terms of the number of algorithms, this study has already considered 9 algorithms. Regarding benchmark functions, this study totally tested 3 groups. The first group consists of unimodal, multimodal, fixed-dimensional, and composition functions with 29 functions. The second group is the CEC-2017 test functions, including 29 functions. Finally, the third group which is the combination of first and second groups include 58 functions. The Friedman statistic requires calculating the mean ranked value. Then, a comparison is needed to review the critical values obtained for the considered significance level ($\alpha = 0.05, 0.1$) with Friedman statistics to see whether the null hypothesis is rejected or not. The formula and explanations can be found in [34] and the critical values are available in appendix B of [35]. For all three groups of benchmarks, the null hypothesis was rejected, showing that there is a significant difference among the performance of the methods in each group.

Further steps are needed to see which algorithms' performance are significantly different than EO and which ones are similar. To this end, we run a post hoc statistical analysis of Bonferroni-Dunn [35], which demonstrates that the performance of two algorithms are significantly different if the difference in average ranking of methods is larger than the critical distance (CD) [34]. Since this study plans to compare the performance of EO with respect to the other algorithms, the control algorithm is chosen to be EO. Fig. 12 shows the average rank of all algorithms in the 3 function groups. In this figure, the horizontal line represents the threshold for the control algorithm (EO). The control algorithm (EO) is able to outperform those algorithms for which their average rank is above the threshold line (i.e., the height of the bars exceeds its corresponding line). For two prevalent significance levels of 0.1 and 0.05, two thresholds are defined and depicted in this figure with dotted and dashed lines, respectively. The threshold line of each group is identified by its color (Group 1 is red, Group 2 is blue, and Group 3 is green). In Group 1,

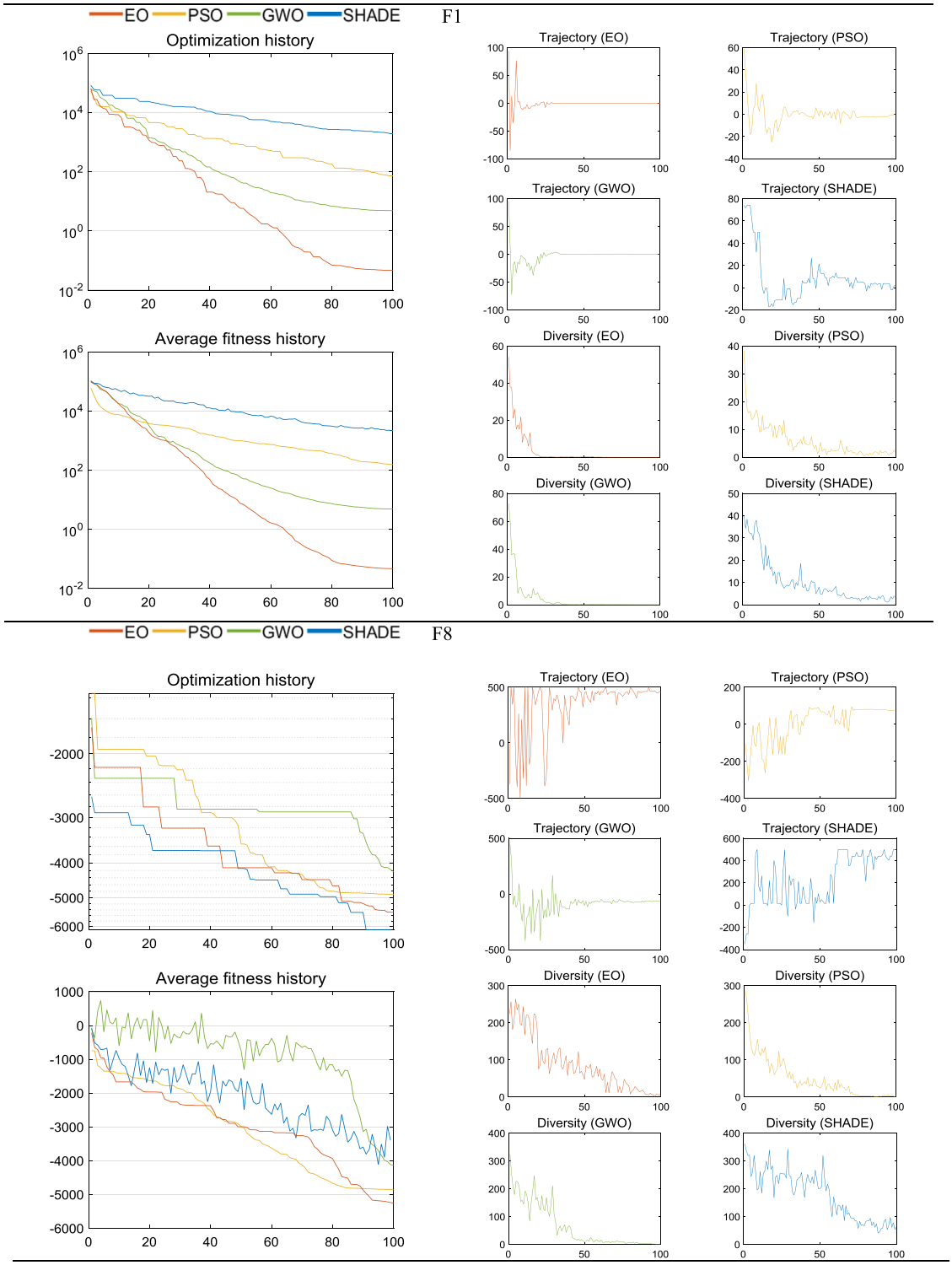

Fig. 9. Comparative qualitative metrics for F1 and F8.

EO outperformed all algorithms, with the lowest average rank of 2.86, and performance was significantly better than PSO, GWO, GA, GSA, SSA, and CMA-ES at both significance levels. In Group 2, EO ranked third after LSHADE-SPACMA and SHADE, and EO significantly outperformed GWO, GA and GSA, SSA, and CMA-ES at both levels of significance. Finally, in Group 3, which compares the performance of each method in all benchmark functions, EO placed third with an average rank of 3.41 and significantly outperformed PSO, GWO, GA, GSA, SSA, and CMA-ES.

One of the weaknesses of Bonferroni-Dunn test is that it is not able to distinguish significant differences of those algorithms that are below the critical line and/or those for which their ranks are close to threshold line (e.g., CMA-ES in Group 2 for $\alpha = 0.05$). Thus, this study also considered another statistical procedure of Holm's [36] to detect which algorithms are better/worse than EO and at which level of significance. Holm's method is a simple and widely applicable multiple test procedure based on the sequential rejective manner. It sorts all algorithms based on their p value and

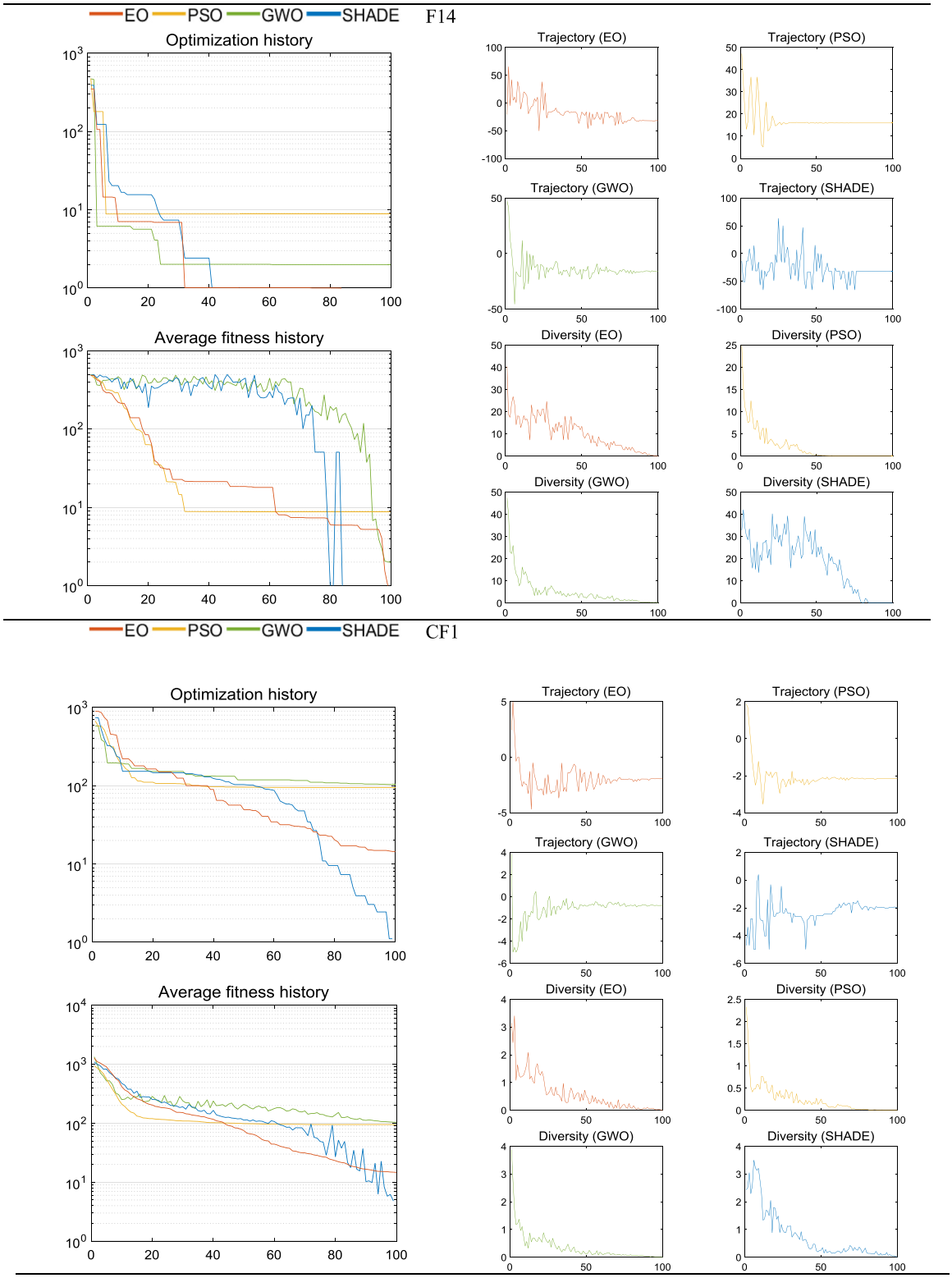

Fig. 10. Comparative qualitative metrics for F14 and CF1.

compares them with $\alpha/k - i$, where α is the significance level, k is the degree of freedom, and i is the algorithm number. The method starts with the most significant p value and sequentially rejects null hypothesis as long as $p_i < \alpha/k - i$. As soon as the method cannot reject the hypothesis, it stops and considers all the remaining hypotheses as accepted. The results for Group 1, shown in Table 4, show that EO significantly outperformed all the algorithms at both significance levels except for SHADE and LSHADE-SPACMA (in other words, the performance of EO, SHADE, and LSHADE-SPACMA are statistically similar to each other in this group). This conclusion is in agreement with previous applications of the Bonferroni–Dunn's test. The results for Group 2, shown in Table 5, show that EO's performance is significantly different from all algorithms except PSO and that it is significantly

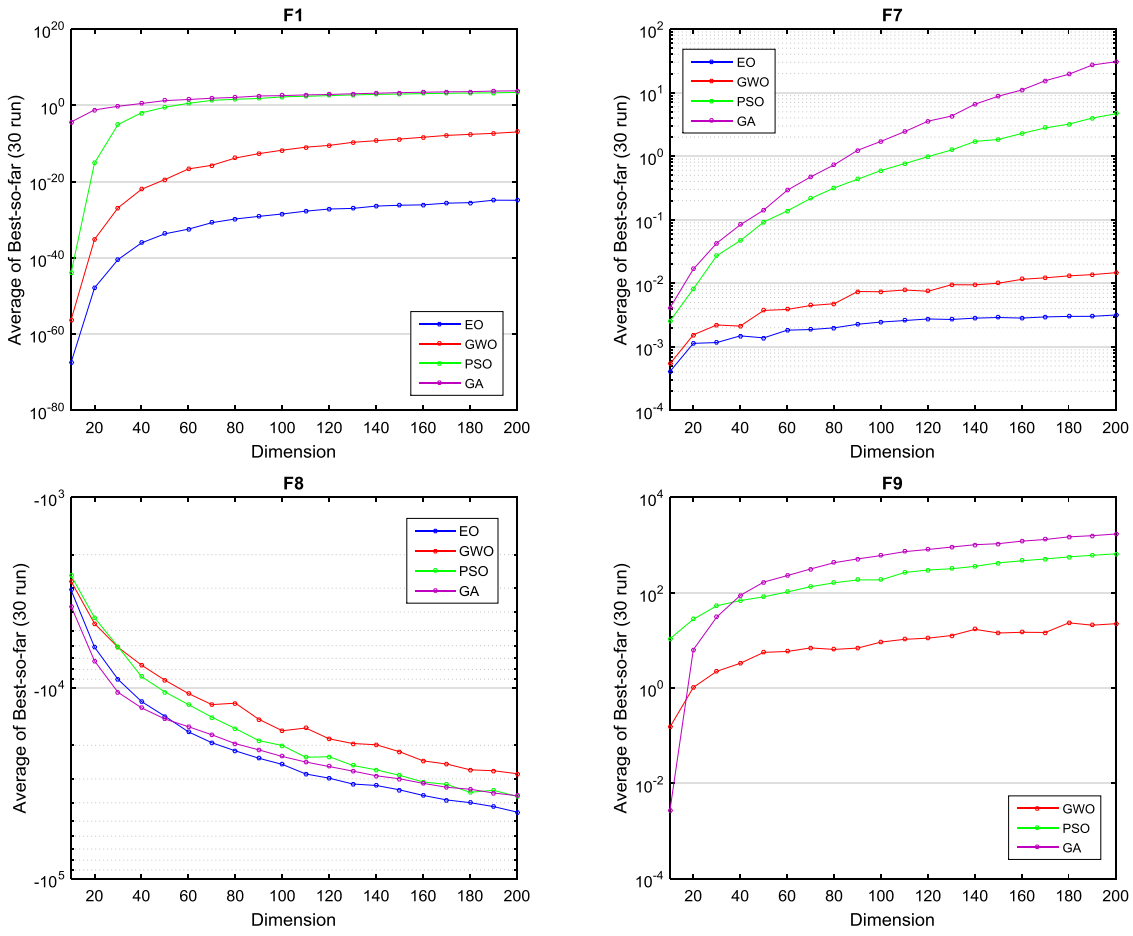

Fig. 11. Robustness comparison of different algorithms in low, high and very high dimensional test cases.

Table 4
Holm's test for group 1 functions (EO is the control algorithm).

EO vs.	Rank	z-value	p-value	α/i (0.05)	α/i (0.1)
GA	6.5862	5.1782	2.40E−06	0.00625	0.0125
GSA	5.6206	3.8357	1.26E−05	0.00714	0.0142
SSA	6.0000	4.3631	1.62E−05	0.00833	0.0166
PSO	5.9482	4.2911	2.22E−05	0.01	0.02
CMA-ES	5.4655	3.6199	2.94E−04	0.0125	0.025
GWO	5.1379	3.1644	0.0016	0.0166	0.0333
LSHADE-SPACMA	3.9137	1.4623	0.1437	0.025	0.05
SHADE	3.4655	0.8390	0.4015	0.05	0.1

Table 5
Holm's test for group 2 functions (EO is the control algorithm).

EO vs.	Rank	z-value	p-value	α/i (0.05)	α/i (0.1)
GSA	7.6896	5.1782	2.40E−06	0.00625	0.0125
GA	6.6896	3.7877	1.52E−04	0.00714	0.0142
LSHADE-SPACMA	1.4482	−3.5000	4.65E−04	0.00833	0.0166
GWO	6.3793	3.3562	7.90E−04	0.01	0.02
SSA	6.1034	2.9726	0.003	0.0125	0.025
SHADE	2.0344	−2.6849	0.0073	0.0166	0.0333
CMA-ES	5.7241	2.4452	0.0145	0.025	0.05
PSO	4.9655	1.3904	0.1644	0.05	0.1

Table 6
Holm's test for group 3 functions (EO is the control algorithm).

EO vs.	Rank	z-value	p-value	α/i (0.05)	α/i (0.1)
GSA	6.655172	6.373797	2.38E−09	0.00625	0.0125
GA	6.637931	6.339894	2.95E−09	0.00714	0.0142
SSA	6.051724	5.187186	2.29E−06	0.00833	0.0166
GWO	5.758621	4.610832	1.94E−05	0.01	0.02
CMA-ES	5.594828	4.288752	2.25E−05	0.0125	0.025
PSO	5.456897	4.017526	5.95E−05	0.0166	0.0333
LSHADE-SPACMA	2.681034	−1.44088	0.1496	0.025	0.05
SHADE	2.75	−1.30527	0.1918	0.05	0.1

better than GSA, GA, GWO, SSA, and CMA-ES at both levels and worse than LSHADE-SPACMA and SHADE at both levels. Finally, in Group 3, shown in Table 6, EO's performance is significantly better than all algorithms at both levels except for SHADE and LSHADE-SPACMA, for which their performance is again similar. As the main conclusion drawn from all functions' statistical analysis in this study, overall, EO performs significantly better than PSO, GWO, GA, GSA, SSA, CMA-ES and performs similar to SHADE and LSHADE-SPACMA.

5. Engineering optimization test problems

This section tests the EO algorithm on three well-known engineering design problems. A simple constraint handling method, static penalty, is applied here to penalize the objective function with a large value if any constraints at any level are violated from its defined bounds. The penalty coefficient should be large enough to properly penalize the objective function in equality/inequality constraints. It is noted that all the engineering test problems are conducted using the same number of iterations (500) and particles (30) as the mathematical functions described previously. All figures and equations of engineering problems are available in the supplementary material.

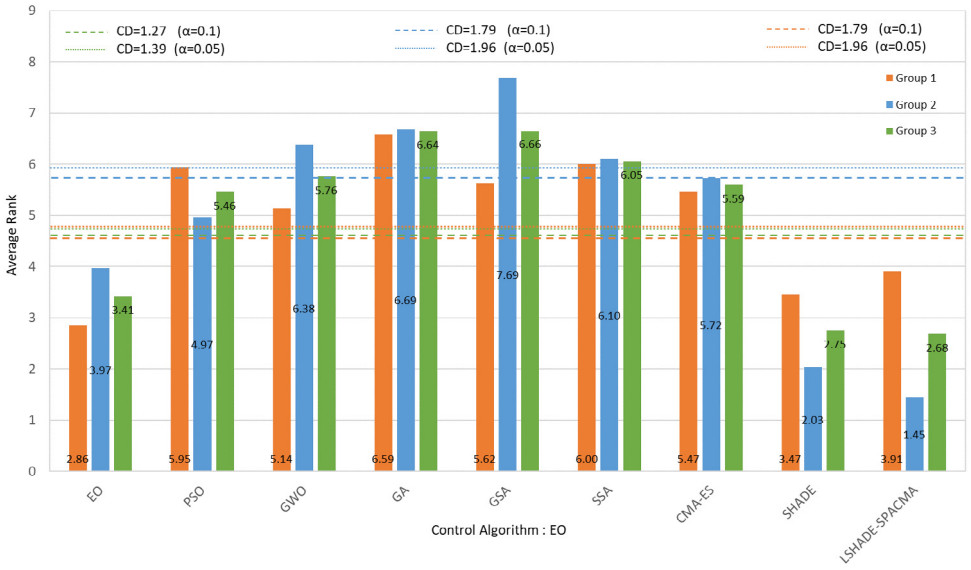

Fig. 12. Bonferroni–Dunn test for different algorithms and benchmark groups with $\alpha = 0.05$ and $\alpha = 0.1$.

Table 7
Optimum results and comparison for the pressure vessel design problem.

Algorithm	$T_s(x_1)$	$T_h(x_2)$	$R(x_3)$	$L(x_4)$	f_{cost}
HGA(2) [37]	1.1250	0.5625	58.1267	44.5941	6832.583
T-Cell [38]	0.8125	0.4375	42.098429	190.787695	6390.554
HGA(1) [37]	0.8125	0.4375	42.0492	177.2522	6065.821
CPSO [39]	0.8125	0.4375	42.091266	176.746500	6061.0777
BFOA [40]	0.8125	0.4375	42.096394	176.683231	6060.460
HAIS-GA [41]	0.8125	0.4375	42.0931	176.7031	6060.367
DTS-GA [42]	0.8125	0.4375	42.097398	176.654047	6059.9463
ES [43]	0.8125	0.4375	42.098087	176.640518	6059.745
CDE [44]	0.8125	0.4375	42.098411	176.637690	6059.7340
EO (Mixed variable)	0.8125	0.4375	42.0984456	176.6365958	6059.7143
EO (Continuous)	0.7781686507	0.3846491672	40.31961921	199.9999933	5885.3279

5.1. Pressure vessel design

This problem is one of the most well-known benchmark design tests with a mixed type of variables (continuous/discrete). A cylindrical pressure vessel capped at both ends with hemispherical heads should work under the pressure of 3000 psi (2.1×10^7 Pa) and a minimum volume of 750 ft^3 (21.24 m^3) according to the ASME boiler code requirement. The total cost of welding, material, and forming define an objective function to be minimized. Variables include the thickness of the shell (T_s), thickness of the head (T_h), the inner radius (R), and the length of the cylindrical section of the vessel (L). Both thickness variables (T_s, T_h) must be integer multiple values of 0.0625 inch, which is the available thickness of rolled steel plates.

Table 7 shows the optimum results obtained by EO and other methods in the literature. Since some researchers considered this case as a continuous problem, the results are presented for EO in both mixed variable and continuous forms and compared only to mixed variable solutions. As Tables 7 and 8 indicate, EO outperformed other methods. The statistical results are presented in Table 8.

Table 8
Statistical results and comparison for the pressure vessel design problem (N. A = not available).

Algorithm	Best	Mean	Worst	Std	Eval, No
HGA(2) [37]	6832.584	7187.314	8012.615	276	80,000
T-Cell [38]	6390.554	6737.065	7694.066	357	80,000
HGA(1) [37]	6065.821	6632.376	8248.003	515	80,000
CPSO [39]	6061.0777	6147.1332	6363.8041	86.4545	200,000
BFOA [40]	6060.460	6074.625	N. A	156	48,000
HAIS-GA [41]	6061.1229	6743.0848	7368.0602	457.99	150,000
DTS-GA [42]	6059.9463	6177.2532	6469.322	130.92	80,000
ES [43]	6059.746	6850.00	7332.87	426	25,000
CDE [44]	6059.7340	6085.2303	6371.0455	43.0130	240,000
EO	6059.7143	6668.114	7544.4925	566.24	15,000

5.2. Welded beam design

The welded beam design is another practical example which has often served as a benchmark case to test different optimization algorithms. It is a cantilever beam welded at one end and subjected to a point load (P) at the other end. The objective is to minimize fabricating cost subjected to shear stress (τ), bending stress (σ), buckling load (P_c), deflection (δ), and other constraints. The variables are thickness of the weld (h), length of welded part of the beam (l), height of the beam (t), and width of the beam (b).

Others that have studied this problem with different algorithms. Table 9 summarizes the optimal results for different algorithms while Table 10 provides the statistical summary of results. The results of the tables indicate that EO outperformed other algorithms with less cost compared to others. EO also obtained the results in a low number of function evaluations and with lower values for standard deviation, mean, and worst solutions compared to most other algorithms.

Table 9
Optimum results and comparison for the welded beam design problem.

Algorithm	$h(x_1)$	$l(x_2)$	$t(x_3)$	$b(x_4)$	f_{cost}
SBM [45]	0.2407	6.4851	8.2399	0.2497	2.4426
BFOA [40]	0.2057	3.4711	9.0367	0.2057	2.3868
SCA [46]	0.2444	6.2380	8.2886	0.2446	2.3854
EA [47]	0.2443	6.2201	8.2940	0.2444	2.3816
T-Cell [38]	0.2444	6.1286	8.2915	0.2444	2.3811
FSA [48]	0.2444	6.1258	8.2939	0.2444	2.3811
IPSO [49]	0.2444	6.2175	8.2915	0.2444	2.3810
DSS-DE [50]	0.2444	6.1275	8.2915	0.2444	2.3810
HSA-GA [51]	0.2231	1.5815	12.8468	0.2245	2.2500
CDE [44]	0.2031	3.5430	9.03350	0.2062	1.7335
CPSO [39]	0.2024	3.5442	9.04821	0.2057	1.7280
TEO [52]	0.2057	3.4723	9.03513	0.2058	1.7253
EO	0.2057	3.4705	9.03664	0.2057	1.7249

Table 10
Statistical results and comparison for the welded beam design problem (N. A = not available).

Algorithm	Best	Mean	Worst	Std	Eval, No
SBM [45]	2.4426	2.5215	2.6315	N. A	19,259
BFOA [40]	2.3868	2.4040	N. A	0.016	48,000
SCA [46]	2.3854	3.2551	6.3996	0.9590	33,095
EA [47]	2.3816	N. A	2.38297	0.00034	28,897
T-Cell [38]	2.3811	2.4398	2.7104	0.09314	320,000
FSA [48]	2.3811	2.4041	2.4889	N. A	56,243
IPSO [49]	2.3810	2.3819	N. A	0.00523	30,000
HSA-GA [51]	2.2500	2.26	2.28	0.0078	26,466
CDE [44]	1.7335	1.768158	1.824105	0.022194	240,000
CPSO [39]	1.7280	1.748831	1.782143	0.012926	200,000
TEO [52]	1.725284	1.768040	1.931161	0.058166	N. A
EO	1.724853	1.726482	1.736725	0.003257	15,000

Table 11
Optimum results and comparison for the spring design problem.

Algorithm	$d(x_1)$	$D(x_2)$	$N(x_3)$	f_{cost}
SI [53]	0.050417	0.321532	13.97991	0.013060
GA(1) [55]	0.051480	0.351661	11.632201	0.012704
CA [54]	0.050000	0.317395	14.031795	0.012721
GSA	0.050276	0.323680	13.525410	0.012702
GA(2) [56]	0.051989	0.363965	10.890522	0.012681
CPSO [39]	0.051728	0.357644	11.244543	0.012674
BFOA [40]	0.051825	0.359935	11.107103	0.012671
CDE [44]	0.051609	0.354714	11.410831	0.012670
SCA [46]	0.052160	0.368159	10.648442	0.012669
HGA [57]	0.051302	0.347475	11.852177	0.012668
T-Cell [38]	0.051622	0.355105	11.384534	0.012665
EO	0.0516199100	0.355054381	11.38796759	0.012666

5.3. Tension/compression spring design

Another well-known benchmark problem is the design of a tension/compression spring with the objective of weight minimization subject to constraints on minimum deflection, shear stress, surge frequency, and some box constraints. The design variables include wire diameter (d), mean coil diameter (D) and number of active coils (N).

Meta-heuristic algorithms [38,40,44,53,54] have been used to solve this problem. Tables 11 and 12 present the optimum and statistical summary results. EO obtained better results compared to the other algorithms except T-cell, which accomplished the problem in 36,000 function evaluations, which is roughly 2.4 times computationally more expensive than EO with 15,000 function evaluations. The statistical results show that even with a smaller number of function evaluations, EO had very competitive statistical results compared to the other algorithms such as PSO, GA, and DE.

Table 12
Statistical results and comparison for the spring design problem (N. A = not available).

Algorithm	Best	Mean	Worst	Std	Eval, No
SI [53]	0.013060	0.015526	0.018992	N. A	20,000
GA(1) [55]	0.012704	0.012769	0.012822	3.93E−05	N.A.
CA [54]	0.012721	0.013568	0.0151156	8.4E−04	50,000
GA(2) [56]	0.012681	0.012742	0.012973	9.5E−05	80,000
CPSO [39]	0.012674	0.012730	0.012924	5.19E−05	200,000
BFOA [40]	0.012671	0.012759	N. A	1.36E−04	48,000
CDE [44]	0.012670	0.012703	0.012790	2.07E−05	240,000
SCA [46]	0.012669	0.012922	0.016717	5.92E−04	25,167
HGA [57]	0.012668	0.013481	0.016155	N.A.	36,000
T-Cell [38]	0.012665	0.013309	0.012732	9.4E−05	36,000
EO	0.012666	0.013017	0.013997	3.91E−04	15,000

6. Conclusion

This paper proposed a novel physics-based optimization algorithm called Equilibrium Optimizer (EO) which was inspired by a generic mass balance equation for a control volume. The design of the EO algorithm includes high exploratory and exploitative search mechanisms to randomly change solutions. Particles with their concentrations are considered as search agents, equivalent to particles and positions in Particle Swarm Optimization (PSO). Concentrations are randomly updated with respect to fit solutions called equilibrium candidates. This random updating along with a properly defined Generation rate term improves EO's exploratory behavior in initial iterations and exploitative search in the final iterations, aiding in local minima avoidance throughout the whole optimization process. Balancing exploration and exploitation provide adaptive values for the controlling parameter and reduces the magnitude of movement for the particles. The efficiency and effectiveness of EO using quantitative and qualitative metrics were validated by testing it on a total of 58 mathematical benchmark functions along with three engineering problems. Comparisons with several well-studied, recent, and high-performance optimization algorithms showed a high effectiveness of the proposed EO algorithm in obtaining optimal or near-optimal solutions with typically higher efficiency (i.e., less computational time or fewer iterations) for the majority of problems investigated. Future studies should work toward developing binary and multi-objective versions of the EO algorithm to further improve performance.

Appendix A

t: iteration, n: Population, d: dimension, c: number of offsprings, m: number of mutated populations, α = coefficient that shows the percentage of sum of offsprings and mutated population to the total number of populations.

PSO complexity

$$O(\text{PSO}) = O(\text{Problem definition}) + O(\text{Initialization}) + O(\text{t}(\text{function evaluation})) + O(\text{t}(\text{Memory saving})) + Ot(\text{Position update})$$

$$O(1) + O(\text{nd}) + O(\text{tcn}) + O(\text{tn}) + O(\text{tnd}) = O(tcn + tnd + tn + td + 1) \cong O(tcn + tnd)$$

$$O(PSO) = O(tcn + tnd)$$

GA complexity

$$O(\text{GA}) = O(\text{Problem definition}) + O(\text{Initialization}) + O(T(\text{function evaluation})) + O(\text{T}(\text{Rolletewheel})) + O(\text{T}(\text{Crossover})) + O(\text{T}(\text{Mutation}))$$

$c + m$ is equal to αn; therefore, $T = \frac{t}{\alpha}$. Using this equation, the updated GA complexity equation is:

$$O(1) + O(\mathrm{nd}) + O(\mathrm{Tcd}) + O(\mathrm{c}) + O(\mathrm{Tcd}) + O(\mathrm{Tmpd}) \cong O(\mathrm{Tmpd} + \mathrm{cd} + Tcn)$$

We consider $\alpha = 1$ Thus T = t and $c + m = n$

$$O(GA) = O(t(c + m)d + tcn) = O(tnd + tcn)$$

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2019.105190.

References

[1] D.G. Luenberger, Linear and Nonlinear Programming, second ed., Addison-Wesley, 1984.
[2] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, U.K, 2004.
[3] M.H. Afshar, A. Faramarzi, Size optimization of truss structures by cellular automata, J. Comput. Sci. Eng. 3 (2010) 1–9.
[4] A. Faramarzi, M.H. Afshar, A novel hybrid cellular automata–linear programming approach for the optimal sizing of planar truss structures, Civ. Eng. Environ. Syst. 31 (2014) 209–228, http://dx.doi.org/10.1080/10286608.2013.820280.
[5] A. Faramarzi, M.H. Afshar, Application of cellular automata to size and topology optimization of truss structures, Sci. Iran. 19 (2012) 373–380, http://dx.doi.org/10.1016/j.scient.2012.04.009.
[6] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.
[7] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[8] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proc. Sixth Int. Symp. Micro Mach. Hum. Sci, 1995 MHS 95, 1995, pp. 39–43, http://dx.doi.org/10.1109/MHS.1995.494215.
[9] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680, http://dx.doi.org/10.1126/science.220.4598.671.
[10] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (2006) 28–39, http://dx.doi.org/10.1109/MCI.2006.329691.
[11] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.
[12] L. Fogel, A. Owens, M. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley, 1966.
[13] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evol. Comput. 11 (2003) 1–18, http://dx.doi.org/10.1162/106365603321828970.
[14] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (2013) 17–35, http://dx.doi.org/10.1007/s00366-011-0241-y.
[15] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191, http://dx.doi.org/10.1016/j.advengsoft.2017.07.002.
[16] A. Kaveh, N. Farhoudi, A new optimization method: Dolphin echolocation, Adv. Eng. Softw. 59 (2013) 53–70, http://dx.doi.org/10.1016/j.advengsoft.2013.03.004.
[17] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, Inform. Sci. 179 (2009) 2232–2248, http://dx.doi.org/10.1016/j.ins.2009.03.004.
[18] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mech. 213 (2010) 267–289, http://dx.doi.org/10.1007/s00707-009-0270-4.
[19] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE Congr. Evol. Comput., 2007, pp. 4661–4667, http://dx.doi.org/10.1109/CEC.2007.4425083.
[20] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (2011) 303–315, http://dx.doi.org/10.1016/j.cad.2010.12.015.
[21] W.W. Nazaroff, L. Alvarez-Cohen, Environmental Engineering Science, first ed., Wiley, New York, 2000.
[22] Z. Guo, Review of indoor emission source models. Part 1. Overview, Environ. Pollut. 120 (2002) 533–549, http://dx.doi.org/10.1016/S0269-7491(02)00187-2.
[23] R.P. Parouha, K.N. Das, A memory based differential evolution algorithm for unconstrained optimization, Appl. Soft Comput. 38 (2016) 501–517, http://dx.doi.org/10.1016/j.asoc.2015.10.022.
[24] T. Beielstein, K.E. Parsopoulos, M.N. Vrahatis, Tuning PSO Parameters Through Sensitivity Analysis, Tech. Rep. Reihe Comput. Intell. CI 12402, Department of Computer Science, University of Dortmund, 2002.
[25] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102, http://dx.doi.org/10.1109/4235.771163.
[26] Novel Composition Test Functions for Numerical Global Optimization - IEEE Conference Publication, 2017, http://ieeexplore.ieee.org/document/1501604/. (Accessed 4 December 2017).
[27] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congr. Evol. Comput, 2013, pp. 71–78, http://dx.doi.org/10.1109/CEC.2013.6557555.
[28] F. van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Inform. Sci. 176 (2006) 937–971, http://dx.doi.org/10.1016/j.ins.2005.02.003.
[29] M. Črepinšek, M. Mernik, S.-H. Liu, Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees, Int. J. Innov. Comput. Appl. 3 (2011) 11–19, http://dx.doi.org/10.1504/IJICA.2011.037947.
[30] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, ACM Comput. Surv. 45 (2013) 35:1–35:33, http://dx.doi.org/10.1145/2480741.2480752.
[31] N.H. Awad, M.Z. Ali, P.N. Suganthan, J.J. Liang, B.Y. Qu, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, in: 2017 IEEE Congr. Evol. Comput. CEC, 2017.
[32] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, P.N. Suganthan, CEC 2017 Special Session on Single Objective Numerical Optimization Single Bound Constrained Real-Parameter Numerical Optimization, in: 2017 IEEE Congr. Evol. Comput. CEC, 2017.
[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[34] S.-H. Liu, M. Mernik, D. Hrnčič, M. Črepinšek, A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model, Appl. Soft Comput. 13 (2013) 3792–3805, http://dx.doi.org/10.1016/j.asoc.2013.05.010.
[35] J.H. Zar, Biostatistical Analysis, Prentice Hall, Englewood Cliffs, 1999.
[36] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.
[37] H.S. Bernardino, H.J.C. Barbosa, A.C.C. Lemonge, L.G. Fonseca, A new hybrid AIS-GA for constrained optimization problems in mechanical engineering, in: 2008 IEEE Congr. Evol. Comput. IEEE World Congr. Comput. Intell, 2008, pp. 1455–1462, http://dx.doi.org/10.1109/CEC.2008.4630985.
[38] V.S. Aragón, S.C. Esquivel, C.A.C. Coello, A modified version of a T-cell algorithm for constrained optimization problems, Internat. J. Numer. Methods Engrg. 84 (2010) 351–378, http://dx.doi.org/10.1002/nme.2904.
[39] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (2007) 89–99, http://dx.doi.org/10.1016/j.engappai.2006.03.003.
[40] E. Montes, B. Ocana, Bacterial foraging for engineering design problems: preliminary results, in: 4th Mex. Congr. Evol. Comput., COMCEV'2008, Mexico, 2008, pp. 33–38.
[41] C.A.C. Coello, N.C. Cortés, Hybridizing a genetic algorithm with an artificial immune system for global optimization, Eng. Optim. 36 (2004) 607–634, http://dx.doi.org/10.1080/03052150410001704845.
[42] C. Coello, E. Montes, Use of dominance-based tournament selection to handle constraints in genetic algorithms, in: Intell. Eng. Syst. Artif. Neural Netw., ANNIE2001, ASME Press, St. Louis, Missouri, 2001, pp. 177–182.
[43] E. Mezura-Montes, C.A.C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (2008) 443–473, http://dx.doi.org/10.1080/03081070701303470.
[44] F. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (2007) 340–356, http://dx.doi.org/10.1016/j.amc.2006.07.105.
[45] S. Akhtar, K. Tai, T. Ray, A socio-behavioural simulation model for engineering design optimization, Eng. Optim. 34 (2002) 341–354, http://dx.doi.org/10.1080/03052150212723.
[46] T. Ray, K.M. Liew, Society and civilization: An optimization algorithm based on the simulation of social behavior, IEEE Trans. Evol. Comput. 7 (2003) 386–396, http://dx.doi.org/10.1109/TEVC.2003.814902.
[47] J. Zhang, C. Liang, Y. Huang, J. Wu, S. Yang, An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization, Appl. Math. Comput. 211 (2009) 392–416, http://dx.doi.org/10.1016/j.amc.2009.01.048.

[48] A.-R. Hedar, M. Fukushima, Derivative-free filter simulated annealing method for constrained continuous global optimization, J. Global Optim. 35 (2006) 521–549, http://dx.doi.org/10.1007/s10898-005-3693-z.
[49] S. He, E. Prempain, Q.H. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Eng. Optim. 36 (2004) 585–605, http://dx.doi.org/10.1080/03052150410001704854.
[50] M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Inform. Sci. 178 (2008) 3043–3074, http://dx.doi.org/10.1016/j.ins.2008.02.014.
[51] S.F. Hwang, R.S. He, A hybrid real-parameter genetic algorithm for function optimization, Adv. Eng. Inf. 20 (2006) 7–21, http://dx.doi.org/10.1016/j.aei.2005.09.001.
[52] A. Kaveh, A. Dadras, A novel meta-heuristic optimization algorithm: Thermal exchange optimization, Adv. Eng. Softw. 110 (2017) 69–84, http://dx.doi.org/10.1016/j.advengsoft.2017.03.014.
[53] T. Ray, P. Saini, Engineering design optimization using a swarm with an intelligent information sharing among individuals, Eng. Optim. 33 (2001) 735–748, http://dx.doi.org/10.1080/03052150108940941.
[54] C.A.C. Coello, R.L. Becerra, Efficient evolutionary optimization through the use of a cultural algorithm, Eng. Optim. 36 (2004) 219–236, http://dx.doi.org/10.1080/03052150410001647966.
[55] C.A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113–127, http://dx.doi.org/10.1016/S0166-3615(99)00046-9.
[56] C.A. Coello Coello, E. Mezura Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inf. 16 (2002) 193–203, http://dx.doi.org/10.1016/S1474-0346(02)00011-3.
[57] H.S. Bernardino, H.J.C. Barbosa, A.C.C. Lemonge, A hybrid genetic algorithm for constrained optimization problems in mechanical engineering, in: 2007 IEEE Congr. Evol. Comput, 2007, pp. 646–653, http://dx.doi.org/10.1109/CEC.2007.4424532.