Engineering Applications of Artificial Intelligence 87 (2020) 103294

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A powerful variant of symbiotic organisms search algorithm for global optimization☆

Emre Çelik

Department of Electrical and Electronics Engineering, Engineering Faculty, Duzce University, 81620, Turkey

ARTICLE INFO

Keywords:
Symbiotic organisms search
Quasi-oppositional based learning
Chaotic theory
Local search
Benchmark function
Engineering design
Global optimization

ABSTRACT

This paper suggests a new variation to the existing symbiotic organisms search (SOS) algorithm developed by simulating three symbiotic strategies of mutualism, commensalism and parasitism used by the organisms. In the revised version called improved SOS (ISOS), the theory of quasi-oppositional based learning is employed during generation of initial population and in the parasitism phase to raise the possibility of getting closer to high-quality solutions. An efficient alternative for parasitism phase is also presented. The two upgraded parasitism strategies avoid the over exploration issue of original parasitism phase that causes unwanted long-time search in the inferior search space as the solution is already refined. To guide the algorithm perform an exhaustive search around the best solution in attempting to further improve the search model of ISOS, a chaotic local search based on the piecewise linear chaotic map is coupled into the proposed algorithm. Twenty-six benchmark functions and three engineering design problems are tested and a contrast with other popular metaheuristics is widely established. Comparative results substantiate the great contribution of proposed ISOS algorithm in solving various optimization problems with superior global search capability and convergence characteristics which render it useful in handling global optimization problems.

1. Introduction

There are several science fields and moments in our daily life that we need optimization for selecting the most reasonable solution from some set of potential alternatives available. In principle, the main purpose of this sort of selection may be to minimize time, cost or effort required while maintaining or even maximizing the desired quality, efficiency or benefit. Thus, the course of optimization can be described as the progress of attaining the situations that minimize or maximize the value of a predefined objective, cost or fitness function. As understood through its description, optimization serves an important goal of avoiding "*waste*" that our modern world is severely suffering from. To deal with this issue and utilize the available sources as efficiently as possible, both academia and industry in various fields have witnessed growing tendency in developing and benefitting from metaheuristic optimization algorithms, which stochastically seek to find a good solution in a given search domain with a superior property of gradient-free mechanism over the gradient-based classical search techniques.

In accordance with the "no free lunch" theorem proposed in Wolpert and Macready (1997), all optimization problems cannot be solved by only one metaheuristic algorithm. This, in other words, states that if a certain metaheuristic algorithm is able to produce satisfactory results for some certain problems, then there is no way to expect this algorithm to perform the same performance for another set of optimization problems. Thus, upgraded approaches are always welcome to cope with specific problems, which is recognized as an inspiration for researchers to propose completely new metaheuristic algorithms as well as enhance the performance of existing ones by hybridizing with other algorithms and/or integrating additional search paradigms into them. In view of this, a large number of well-recognized algorithms have appeared in this field, some of which can be counted as genetic algorithm (GA) (Holland, 1992), particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), gravitational search algorithm (GSA) (Rashedi et al., 2009), artificial bee colony (ABC) (Karaboga and Basturk, 2007), grey wolf optimization (GWO) (Mirjalili et al., 2014), stochastic fractal search (SFS) (Salimi, 2015), symbiotic organisms search (SOS) (Cheng and Prayogo, 2014), moth-flame optimization (MFO) algorithm (Mirjalili, 2015), squirrel search algorithm (SSA) (Jain et al., 2019), birds foraging search (BFS) (Zhang et al., 2019), and so forth. The second category for further improving the search capability of algorithms includes the hybrids such as extremal multiobjective genetic algorithm (EMOGA) (Pistolesi et al., 2018), PSO-local search (Wu et al., 2014), hybrid big bang–big crunch (HBB–BC) algorithm (Sedighizadeh et al., 2017), hybrid GA and bacterial foraging (BF) approach (Kim

☆ 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.engappai.2019.103294.

E-mail address: emrecelik@duzce.edu.tr.

https://doi.org/10.1016/j.engappai.2019.103294
Received 16 April 2019; Received in revised form 7 July 2019; Accepted 8 October 2019
Available online xxxx

et al., 2007), hybrid stochastic fractal search plus pattern search (hSFS-PS) (Padhy and Panda, 2017), GA based simulated annealing (GASA) approach (Kaplan and Çelik, 2018), SOS algorithm coupled with SA technique (hSOS-SA) (Çelik and Öztürk, 2018b), and adaptive PSO combined with feed-forward back-propagation learning algorithm (APSO-FFBP) (Karkheiran et al., 2019). Our literature inspection also points out that inserting the knowledge of quasi-oppositional based learning (QOBL) and chaotic local search (CLS) into metaheuristics has received considerable attention recently in the hoping of gaining the algorithms' highest potential (Xiang et al., 2007; Saha and Mukherjee, 2016, 2018; Guhaa et al., 2017; Shiva et al., 2015; Mirjalili and Gandomi, 2017; Roy and Bhui, 2013; Truong et al., 2019).

Among the developing approaches that we see in the field of metaheuristic algorithms, SOS algorithm first introduced by Cheng and Prayogo (2014) for complex numerical optimization is recognized as an efficient optimizer in literature. It is a population-based algorithm inspired by the simulation of symbiotic relationships existing in distinct organisms that cooperate to stay alive in the ecosystem. The three phases of the algorithm, i.e. mutualism, commensalism and parasitism, are easy to code with simple equations and conditional statements. Moreover, other important advantage of the basic SOS is that it requires the tuning of only two common parameters (number of organisms and iteration number), which raises the algorithm robustness with respect to random initialization. This property also minimizes the time-consuming work of setting algorithm-specific parameters and the risk of degraded search performance owing to bad parameter selection. According to the comparative results presented in the original study (Cheng and Prayogo, 2014), the superior performance of SOS was shown over its competitors in solving various benchmark functions and engineering design problems. This outcome along with the above-said features has made the SOS algorithm very popular among researchers, which accordingly yields to a great number of research works applying SOS and its modified versions to a variety of engineering problems like optimizing the speed drive efficiency of a DC servo system (Çelik and Öztürk, 2018a), PID controller design for an automatic voltage regulator system (Çelik and Öztürk, 2018b; Çelik and Durgut, 2018), vehicle routing problem (Yu et al., 2017), structural design optimization (Tejani et al., 2016) and optimal power flow problem (Saha et al., 2019). Nonetheless, the trade-off between exploration and exploitation capabilities, which are essential merits required for the success of a metaheuristic algorithm, is not balanced well in the case of SOS because it does not have tuning parameters that help the algorithm preserve its balance in exploring and exploiting the given search space. Thus, the basic SOS is subject to being trapped in local optima and premature convergence. This has been addressed by a few studies (Saha and Mukherjee, 2016, 2018; Guhaa et al., 2017) in literature by incorporating the inexpensive paradigms into the algorithm such as QOBL and CLS, serving as a bridge that increases the chance of achieving a more promising solution with a speedier convergence rate than the case without using such approaches.

On the other hand, there is another concern corresponding to the parasitism phase of SOS algorithm. This phase is very important for the algorithm to satisfy its diversification or exploration property, which protects SOS from local minima stagnation before sampling the entire search surface. As the **Parasite_Vector** in this phase is generated by randomly replacing some components of an organism with random numbers, the process becomes useless as the algorithm proceeds and the solution is being refined. In other words, toward the end of the optimization progress, the parasitism phase offers over exploration causing unnecessary computational time by performing too explorative search outside the promising region found so far, in which situation the generated new solutions are not accepted. Thus, upgrading the parasitism phase to save the computational time without compromising on the solution accuracy is of great interest, and forms one of the motivations of the current paper.

Considering the above deficiencies of the original SOS technique and inspired by the impressiveness of QOBL and CLS paradigms, the current paper proposes an improved SOS (ISOS) algorithm that combines the strength of QOBL and CLS in an innovative way. After obtaining quasi-opposite population initially, QOBL is, for the first time, utilized in the parasitism phase and generation jumping is skipped in contrast to the reported works for the concern of computation cost. An efficient alternative for parasitism phase is also presented. Next, in order to intensify the search process towards the neighborhood of the global best organism, the piecewise linear chaotic map (PWLCM) is employed to induce chaotic search with a better chaotic behavior and higher speed. The impressive supremacy of proposed ISOS algorithm over other popular algorithms is thoroughly illustrated on a wide set of optimization problems including 26 well-known test functions and three practical engineering design problems (tension/compression spring, pressure vessel and PID controlled automatic voltage regulator designs). An important contribution of the present work is that it does not undermine the simplicity advantage of the original SOS in that the number of tuning parameters is increased only by one, i.e. iteration number of chaotic map. Another remarkable merit of ISOS scheme is that it could substantially improve both solution quality and convergence profile in most of the cases by capturing a good trade-off between exploration and exploitation.

2. Overview of SOS algorithm

One of the available metaheuristics that researchers worldwide are keen on using is SOS algorithm, which was introduced by Cheng and Prayogo in 2014 (Cheng and Prayogo, 2014). Like many other metaheuristic algorithms, SOS is population-based, nature-inspired and benefits from randomness to some degree. In the algorithm, the natural phenomenon of reliance-based interaction known as *symbiosis* is exploited, which is a need for organisms in nature to stay alive in the ecosystem. Two types of symbiotic relationships may exist between any two distinct organisms, either compulsory or facultative. In the first case, the survival of two species depends on each other while in the latter case two species can nonessentially cohabitate in a mutually beneficial relationship.

In SOS, the search process is initialized by a random population of N organisms. Then population members are improved by using three realistic symbiotic phases of mutualism, commensalism and parasitism. As the iteration of SOS continues, the above three phases are executed subsequently where the solution generated from each interaction is unconditionally accepted if its fitness value is better than its pre-interaction fitness and otherwise it is rejected. In the following sub-sections, the so-called phases are briefly enumerated.

2.1. Mutualism phase

This SOS phase imitates the mutualistic relationship which is beneficial for both organisms, i.e. each organism is positively affected from the other's activity. Assuming X_i as the ith organism of the ecosystem and X_j randomly selected organism where $j \neq i$, both organisms interacts in a mutualistic sense to increase their existence chance in the ecosystem. As the result of this interaction, new trial solutions X_{inew} and X_{jnew} are calculated using Eqs. (1) and (2) and they replace X_i and X_j if their fitness value is better.

$$X_{inew} = X_i + rand\,(0,\ 1) \times \left(X_{best} - Mutual\_Vector \times BF1\right) \tag{1}$$

$$X_{jnew} = X_j + rand\,(0,\ 1) \times \left(X_{best} - Mutual\_Vector \times BF2\right) \tag{2}$$

$$Mutual\_Vector = \frac{X_i + X_j}{2} \tag{3}$$

where $rand\,(0,\ 1)$ returns random numbers from the uniform distribution in the interval [0, 1], X_{best} is the ecosystem's best organism, $BF1$ and $BF2$ are benefit factors randomly assigned as either 1 (partial benefit) or 2 (full benefit), determining the degree of benefit to each organism. Apparently, if one organism benefits from the interaction

in full, i.e. $BF1 = BF2 = 2$, the terms $Mutual\_Vector \times BF1$ and $Mutual\_Vector \times BF2$ in Eqs. (1) and (2) will yield to a great diversity for X_{inew} and X_{jnew} compared to the case of $BF1 = BF2 = 1$. Thus, it can be concluded that low value of benefit factor allows an exploitative search in a small area and large value of benefit factor guides an explorative search over the unexplored regions of the search space. So, the balance between exploration and exploitation in this phase depends largely on the random values of benefit factors.

2.2. Commensalism phase

In commensalism phase, one organism gains benefit while the other organism is not influenced from this engagement neither negatively nor positively. Analogous to the mutualism phase, X_j is randomly picked up from the ecosystem. Herein, X_i is the organism that aims to benefit from the interaction while X_j is the neutral one insensitive to this kind of relationship. The new trial solution X_{inew} is calculated in Eq. (4) and the algorithm is forwarded with X_{inew} if it is better than X_i.

$$X_{inew} = X_i + rand\,(-1,\ 1) \times \left(X_{best} - X_j\right) \tag{4}$$

It is noticeable from Eq. (4) that the new trial organism is obtained based on the difference $X_{best} - X_j$ multiplied by a random number, which is now in a wider range of −1 and 1 to widen the search space in comparison with the case using $rand\,(0,\ 1)$.

2.3. Parasitism phase

Parasitism is a kind of symbiotic relationship that one organism, the parasite, adopts for sustenance by benefitting from another organism, the host, causing it some harm. In SOS, the artificial parasite organism called **Parasite_Vector** is created by duplicating and altering some random components of X_i with a random number in the lower LB and upper UB search boundaries as shown mathematically in Eq. (5). Then a different organism X_j in the ecosystem is assigned as a host organism to the parasite. Both organisms try to exclude each other from the ecosystem and the one with a better fitness value will kill the other one and conquer its position in the ecosystem.

$$Parasite\_Vector = \begin{cases} X_i^d & if\ rand\,(0,\ 1) < rand\,(0,\ 1) \\ LB + rand\,(0,\ 1) \times (UB - LB) & else \end{cases} \tag{5}$$

where $Parasite\_Vector = \left[X^1, X^2, \ldots, X^D\right]$, and D means the number of design variables.

Parasitism phase introduces random variations in the ecosystem with an aim to protect organisms from local minima stagnation, and thus it plays a key element in satisfying the global search performance or the exploration capability of the algorithm. However, the level of exploration becomes a sticky issue as the algorithm already converges to a promising region after a number of generations in which case most of the generated solutions will be rejected owing to poor solution quality of theirs. This leads to higher computation burden and sluggish convergence mobility. Encouraged by this motive, two effective parasitistic strategies are suggested in the current paper to save the computation time while still maintaining good solution accuracy.

3. The improved symbiotic organisms search algorithm (ISOS)

As understood in the above discussion, the balance between exploration (diversification) and exploitation (intensification) properties must be satisfied in order for most of the metaheuristic algorithms to attain a desirable performance, and it was shown that this balance is not well established in the case of SOS owing to the lack of tuning parameters that direct the search process. In addition, over exploration issue of the SOS's parasitism phase is a matter of concern because it is switched to an inefficient time-consuming process after a certain number of generations. In order to handle these deficiencies, two strategies (QOBL and CLS) and a new scheme for parasitism phase are incorporated into the original SOS protocol, which are the foundation of the current paper. The modifications are described in following subsections.

3.1. Quasi-oppositional based learning

The concept of opposition-based learning (OBL) was developed by Tizhoosh in 2005 for machine intelligence (Tizhoosh, 2005). Following its first successful application to differential evolution algorithm (Rahnamayan et al., 2008a), OBL gains wide acceptability among the researchers in the field of computational intelligence. The philosophy of OBL is to contribute to solution accuracy and convergence speed by comparing the fitness of a candidate solution to its opposite and choosing the better one in the population. It is recognized from the literature (Rahnamayan et al., 2008b) that opposite population has a higher chance to evolve quickly than a population not using OBL. Later on, the theory of oppositional learning has been further expanded to quasi-oppositional based learning, which theoretically demonstrates that quasi-opposite points are more likely to be closer to the solution than an opposite number (Rahnamayan et al., 2007). Definitions of opposite and quasi-opposite points in one and multi-dimensional space are given in the following.

3.1.1. Opposite number

If X is a real number in 1-dimensional search space bounded by $[a, b]$, its opposite, OX, is given by

$$OX = a + b - X \tag{6}$$

where X is a candidate/estimated solution to given problem, a and b are the minimum and maximum search limits. Obviously, OX is generated at the mirror position of X with respect to the center of the search domain. In the case of D-dimensional search space, the above definition can be easily stated by

$$OX^j = a^j + b^j - X^j \tag{7}$$

where $j = 1, 2, \ldots,\ D$ and $X = \left[X^1, X^2, \ldots, X^D\right]$

3.1.2. Quasi-opposite number

The quasi-opposite number, QOX, is the number picked up randomly between the center of the search space $c = (a + b)/2$ and the opposite number OX, and it is mathematically expressed by

$$QOX = rand\left(\frac{a+b}{2}, OX\right) \tag{8}$$

In Eq. (8), the function $rand\,(\bullet)$ generates a random number uniformly distributed between c and OX. Similarly, the quasi-opposite point $QOX = \left[QOX^1, QOX^2, \ldots, QOX^D\right]$ for D-dimensional search space may be stated by

$$QOX^j = rand\left(\frac{a^j + b^j}{2}, OX^j\right) \tag{9}$$

3.1.3. Utilization of QOBL

Our literature inspection points out that the general trend in exploiting the benefits of quasi-opposite numbers is to utilize QOBL in the population initialization and generation jumping. In the first case, initial population including N organisms is generated randomly within search boundaries. After quasi-opposite population is acquired using Eq. (9), the fittest N organisms are chosen as initial population from the combination of X and QOX ($X \cup QOX$). The pseudo code for obtaining quasi-opposite population is given in Algorithm 1.

Algoritm 1: Pseudo code for obtaining quasi-opposite population

for $i = 1: N$
  for $j = 1{:}D$
    $OX_i^j = a^j + b^j - X_i^j$
    $c^j = (a^j + b^j)/2$
    if $OX_i^j < c^j$
      $QOX_i^j = c^j + \left(OX_i^j - c^j\right) \times r$ % $r \in [0, 1]$
    else
      $QOX_i^j = OX_i^j + \left(c^j - OX_i^j\right) \times r$
    endif
  endfor
endfor

In the second case, QOBL is applied to the evolving population depending upon the jumping probability (J_r, jumping rate) just after the main algorithm procedures are finished. In this sense, Algorithm 1 is called in each iteration as long as J_r is greater than a random number, and the algorithm is forwarded with the fittest N organisms selected between the current population and newly generated quasi-opposite population. This process of jumping to a new candidate solution is removed in the proposed algorithm to reduce computational time and number of fitness function evaluation (NFFE). In lieu of this, the knowledge of QOBL is utilized, for the first time, in the creation of **Parasite_Vector** using the following expressions.

$$OX_{best} = a + b - X_{best} \tag{10}$$

$$Parasite\_Vector = QOX_{best} = rand\left(\frac{a+b}{2}, OX_{best}\right) \tag{11}$$

As seen, **Parasite_Vector** is created on the basis of X_{best} being the best organism in the ecosystem characterizing the greatest level of adaption. We look after the quasi-opposite of X_{best} as it is expected to have a higher chance of being closer to the optimal solution than both OX_{best} and X_{best} as suggested by the theory of QOBL. However, since X_{best} is used in Eq. (10), the modified parasitism phase leads to a localized search in the quasi-oppositional domain of X_{best}, which often expedites the convergence, but deteriorates the global search capability or exploration ability of the original parasitism phase. Considering this, another simple and effective alternative for creating **Parasite_Vector** is provided by

$$Parasite\_Vector = \begin{cases} X_m^d & if\ rand\ (0,\ 1) < rand\ (0,\ 1) \\ X_n^d & else \end{cases} \tag{12}$$

The new candidate solution **Parasite_Vector** in Eq. (12) is a combination of random elements of two organisms X_a and X_b randomly selected from the ecosystem, such that $m \neq n$. Such modification is found to yield a better exploration of the search space. **Parasite_Vector** in the proposed ISOS algorithm is generated relied on the two rules in Eqs. (11) and (12) with a probability of 0.5. The fitness of **Parasite_Vector** is compared to that of a different organism X_j in the ecosystem and if **Parasite_Vector** has a better fitness value, then it replaces X_j in the ecosystem. As a result, the proposed parasitism phase resolves the over exploration issue of its original counterpart and saves the computational time while at same time providing an improvement in terms of solution accuracy and convergence speed.

3.2. Chaotic local search

Chaos is essentially a randomness generated by deterministic systems and its performance is highly dependent on the choice of initial value as it determines the chaotic orbit (Xiang et al., 2007; Saha and Mukherjee, 2018). The chaos theory, with the properties of simplicity, stochasticity and ergodicity, has been a popular and fruitful paradigm integrated into population-based algorithms for improving search capability and evading from local optima stagnation (Xiang et al., 2007; Saha and Mukherjee, 2016, 2018; Mirjalili and Gandomi, 2017; Alatas et al., 2009; Gandomi et al., 2012; Coelho and Mariani, 2012; Güvenç et al., 2018). As such, in order to better balance exploration and exploitation, CLS is subsequently invoked after the above-mentioned algorithm procedures. This final version of ISOS algorithm is a sort of two-stage optimization technique as the proposed SOS and chaotic search cooperate and are commutated to each other in each iteration.

As the chaotic map to generate chaotic changes, piecewise linear chaotic map is preferred to the well-known logistic map in the current study because PWLCMs are proved to be computationally more effective as well as better dynamical behavior. More importantly, they were shown to be ergodic and have uniform invariant density function on their definition intervals (Xiang et al., 2007). The PWLCM may be mathematically expressed by Eq. (13) (Saremi et al., 2014).

$$x_{i+1} = \begin{cases} \dfrac{x_i}{P} & 0 \le x_i < P \\ \dfrac{x_i - P}{0.5 - P} & P \le x_i < 0.5 \\ \dfrac{1 - P - x_i}{0.5 - P} & 0.5 \le x_i < 1 - P \\ \dfrac{1 - x_i}{P} & 1 - P \le x_i < 1 \end{cases}, \quad P = 0.4 \tag{13}$$

The distribution of x for this map with $P = 0.4$ is pictured in Fig. 1, where $x_0 = 0.3$ and $x_0 = 0.7$, respectively. As seen, the PWLCM exhibits chaotic behavior in the interval of $(0,\ 1)$. It is also noticeable from Fig. 1 that the values ranged in this interval have almost similar occurrence frequency, which signifies that when this type of map is utilized in chaotic search, ranges of search space have favorably identical possibility to be searched.

Instead of applying chaotic search over all organisms of the population which consumes relatively more time, we implement CLS on the global best particle only because the region around there could be the most encouraging. The chaotic variable produced by PWLCM is transformed back to the variable space using Eq. (14).

$$u_{i+1} = X_{best} + \left(x_{i+1} - 0.5\right)\left(X_m - X_n\right) \tag{14}$$

where u_{i+1} is the newly obtained organism at $(i+1)$th iteration of CLS scheme, x_{i+1} is a chaotic number in the range of $(0,\ 1)$, X_{best} is the best organism of proposed SOS algorithm, X_m and X_n are two organisms randomly picked up from the entire ecosystem. The chaotic variable is initially generated at random ($x_0 = rand\ (0,\ 1)$) whose superiority was already addressed in Xiang et al. (2007). A fitness comparison between u_{i+1} and X_{best} is established in every iteration of chaotic local search, and the one offering a better fitness function value is considered as the best organism.

It should be emphasized that in the initialization mode of the proposed ISOS algorithm, since X_m and X_n are quite disparate from each other lying at different portions of the search space, the term $\left(X_m - X_n\right)$ results in a larger diversity for the chaotic search. As the ISOS progresses, X_m and X_n begin to resemble each other and the term $\left(X_m - X_n\right)$ converges zero, which helps chaotic search area to shrink naturally. That said, the algorithm explores at the beginning and then exploits toward the end of the evolutionary process in the neighborhood of the global best solution. This in turn allows a direct and intensified search for better solution quality with substantial improvement on the convergence speed. Unlike the reported works (Xiang et al., 2007; Saha and Mukherjee, 2016, 2018), the introduced CLS technique eliminates the use of additional parameters such as chaotic search radius r and shrinking coefficient δ which are tunable parameters set for decreasing the range of chaotic search. Herein, a possible drawback of Eq. (14) is that if the initial ecosystem is generated within a smaller domain, the term $\left(X_m - X_n\right)$ will not provide the chaotic search with the necessary diversity at the beginning of the search process, in which condition the substituted search effort cannot contribute to the exploration of the ISOS algorithm. This may come with a cost

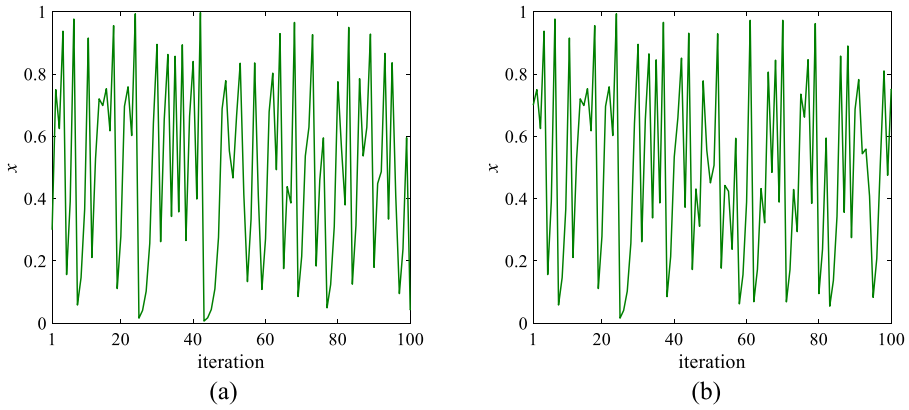

Fig. 1. Visualization of PWLCM with $P = 0.4$ under different start points (a) $x_0 = 0.3$ (b) $x_0 = 0.7$.

Table 1
The details of benchmark test functions with two dimensions.

Function	Range	Type	D	Formulation	Min
Beale (f_1)	[−4.5, 4.5]	UN	2	$f_1(u) = \left(1.5 - u_1 + u_1u_2\right)^2 + \left(2.25 - u_1 + u_1u_2^2\right)^2 + \left(2.625 - u_1 + u_1u_2^3\right)^2$	0
Easom (f_2)	[−100, 100]	UN	2	$f_2(u) = -\cos\left(u_1\right)\ \cos\left(u_2\right)\exp\left[-\left(u_1 - \pi\right)^2 - \left(u_2 - \pi\right)^2\right]$	−1
Matyas (f_3)	[−10, 10]	UN	2	$f_3(u) = 0.26\left(u_1^2 + u_2^2\right) - 0.48u_1u_2$	0
Bohachevsky1 (f_4)	[−100, 100]	MS	2	$f_4(u) = u_1^2 + 2u_2^2 - 0.3\cos\left(3\pi u_1\right) - 0.4\cos\left(4\pi u_2\right) + 0.7$	0
Booth (f_5)	[−10, 10]	MS	2	$f_5(u) = \left(u_1 + 2u_2 - 7\right)^2 + \left(2u_1 + u_2 - 5\right)^2$	0
Michalewicz2 (f_6)	[0, π]	MS	2	$f_6(u) = -\sum_{i=1}^{D}\sin\left(u_i\right)\left[\sin\left(iu_i/\pi\right)\right]^{20}$	−1.8013
Schaffer (f_7)	[−100, 100]	MN	2	$f_7(u) = 0.5 + \dfrac{sin^2\left(\sqrt{u_1^2 + u_2^2}\right) - 0.5}{\left[0 + 0.001\left(u_1^2 + u_2^2\right)\right]^2}$	0
Six Hump Camel Back (f_8)	[−5, 5]	MN	2	$f_8(u) = 4u_1^2 - 2.1u_1^4 + \frac{1}{3}u_1^6 + u_1u_2 - 4u_2^2 + 4u_2^4$	−1.0316
Bohachevsky2 (f_9)	[−100, 100]	MN	2	$f_9(u) = u_1^2 + 2u_2^2 - 0.3\cos\left(3\pi u_1\right)\left(4\pi u_2\right) + 0.3$	0
Bohachevsky3 (f_{10})	[−100, 100]	MN	2	$f_{10}(u) = u_1^2 + 2u_2^2 - 0.3\cos\left(3\pi u_1 + 4\pi u_2\right) + 0.3$	0
Shubert (f_{11})	[−10, 10]	MN	2	$f_{11}(u) = \left(\sum_{i=1}^{5} i\cos\left(i+1\right)u_1 + i\right)\left(\sum_{i=1}^{5} i\cos\left(i+1\right)u_2 + i\right)$	−186.73

D: Dimension, U: Unimodal, M: Multimodal, N: non-separable, S: Separable.

of needless fitness evaluations and time consumption in the bad search space. Therefore, to tackle this limitation, random ecosystem should be initiated from a wider range as much as possible in the sense that initial individuals are located in different points of a given search space. The detailed pseudo code describing the procedures of ISOS is given in Algorithm 2. At the initial stage of Algorithm 2, random ecosystem is generated from the standard uniform distribution and its corresponding quasi-opposite is created using Algorithm 1. An elitist selection strategy is employed to elect the fittest individuals from the union of the original and quasi-opposite populations. Afterwards, the iterative phases of mutualism, commensalism and modified parasitism of SOS algorithm is executed with the first *for* loop to discover the best solution over the whole ecosystem. Chaotic search is then activated for a number of iterations to search for better results by its ergodicity in a search space shrinking around the best solution.

4. Empirical results

The searching performance of ISOS algorithm is thoroughly identified in the following two subsections. The first one contains 26 benchmark functions used by many researchers, and three kinds of engineering design problems are employed in the second one. For each problem, ISOS algorithm is compared to popular techniques in the respective field to verify its results.

4.1. Analytical benchmark problems

To justify the performance of proposed ISOS algorithm, twenty-six prevalent benchmark test functions, described in Tables 1–3 (Saha and Mukherjee, 2018), are employed first in this section. Table 1 contains two-dimensional functions ($f_1 - f_{11}$) and Table 2 comprises four (f_{12}), five (f_{13}) and ten-dimensional (f_{14}, f_{15}) functions, respectively. The rest of the test functions are all thirty-dimensional and included in Table 3. In these tables, D signifies problem dimension and the "Type" column shows the characteristic property of each function such that U and M signify unimodal and multimodal, while N and S denote non-separable and separable. The program code of ISOS algorithm is written in Matlab 9.3.0 (R2017b) program installed on a computer with Intel® Core™ i5, 3.0 Ghz CPU and 8.0 GB memory.

For highlighting the contribution of our proposal, obtained numerical results are also compared to those offered by a recent study (Saha and Mukherjee, 2018) that suggests the use of chaotic SOS (CSOS) algorithm, which results in superior performance than the original SOS. Thus, only the numerical values pertaining to CSOS are reported here for comparison purposes.

The comparison of tuning parameters used in the algorithms is presented in Table 4. Except for the ecosystem size and maximum number of fitness function evaluations, which are set to 50 and 500,000 in two algorithms, only one additional control parameter for chaotic search iteration number needs to be tuned in ISOS. Nonetheless, three more control parameters, such as chaotic search radius, shrinking coefficient and chaotic search iteration number should be determined carefully in presence of CSOS. Thereby, from the perspectives of algorithm simplicity and parameter dependency, ISOS is contemplated to be better than CSOS.

Algoritm 2: Pseudo code for the description of ISOS algorithm

Declare fitness function in D-dimensional search space $f(u); u = [u^1, u^2, .., u^D]$
Generate an initial random ecosystem X of N organisms according to the standard uniform distribution
Use Algorithm 1 to obtain quasi-opposite population QOX
Filter the fittest N organisms from $X \cup QOX$ to form initial population
Set $num\_iter = 0$ *% num_iter is iteration count*
while $num\_iter < max\_iter$ *% max_iter is maximum iteration number*
% SOS algorithm
for $i = 1: N$
Identify best organism X_{best} in ecosystem
Employ mutualism phase
Employ commensalism phase
% Modified parasitism phase
if $rand < 0.5$
Create a **Parasite_Vector** using Eq. 11
else
Create a **Parasite_Vector** using Eq. 12
endif
Accept **Parasite_Vector** to replace randomly selected organism X_j, if it is better than X_j
endfor

% Chaotic local search
Identify best organism X_{best} in ecosystem
Initialize chaotic variable randomly $x_0 = rand(0, 1)$
for $k = 1: K$ *% k is maximum iteration number for chaotic search*
Set x_{i+1} chaotically based on PWLCM by Eq. 13
Pick up two random organisms from ecosystem X_m and X_n, with $m \neq n$
Transform x_{i+1} back to the variable space using the below equation
$u_{i+1} = X_{best} + (x_{i+1} - 0.5)(X_m - X_n)$
Accept u_{i+1} to replace X_{best}, if it is better than X_{best}
endfor *% End of chaotic local search*
$num\_iter = num\_iter + 1$
endwhile *% End of main while looping*

Table 2
The details of benchmark test functions with four, five and ten dimensions.

Function	Range	Type	D	Formulation	Min
Colville (f_{12})	[−10, 10]	UN	4	$f_{12}(u) = 100(u_1^2 - u_2)^2 + (u_1 - 1)^2 + (u_3 - 1)^2 + 90(u_3^2 - u_4)^2 + 10.1\left((u_2 - 1)^2 + (u_4 - 1)^2\right) + 19.8(u_2 - 1)(u_4 - 1)$	0
Michalewicz5 (f_{13})	[0, π]	MS	5	$f_{13}(u) = -\sum_{i=1}^{D} \sin(u_i)\left[\sin(iu_i^2/\pi)\right]^{20}$	−4.6877
Zakharov (f_{14})	[−5, 10]	UN	10	$f_{14}(u) = \sum_{i=1}^{D} u_i^2 + \left(\sum_{i=1}^{D} 0.5iu_i\right)^2 + \left(\sum_{i=1}^{D} 0.5iu_i\right)^4$	0
Michalewicz10 (f_{15})	[0, π]	MS	10	$f_{15}(u) = -\sum_{i=1}^{D} \sin(u_i)\left[\sin(iu_i^2/\pi)\right]^{20}$	−9.6602

D: Dimension, U: Unimodal, M: Multimodal, N: non-separable, S: Separable.

4.1.1. Results for two-dimensional problems

The results of applying ISOS algorithm to the benchmark functions in the first group ($f_1 - f_{11}$) are delineated in Table 5 along with the results reported in Saha and Mukherjee (2018). Due to the stochasticity of evolutionary algorithms, the algorithm is subsequently run 100 times under random initial conditions and the results are then averaged. As per the performance merits, the mean and standard deviation of the best solutions found at the last iteration, the corresponding number of fitness function evaluations and the execution time are provided. Moreover, the algorithms are ranked from the smallest mean solution to the highest one. Their average ranks are reported and the final ranks are designated as the overall rank. In all the reported tables, results of interests are indicated by bold faces in their corresponding section. It is noticeable from Table 5 that for the functions f_2, f_6, f_8 and f_{11}, both ISOS and CSOS have the same results of *mean* and *stDev*. In any of the other test functions, ISOS have favorably achieved the global minimum in each run that can be proved by the observation of *stDev* equal to zero. Regarding the convergence mobility, ISOS outperforms CSOS as it needs significantly fewer fitness function evaluations to converge fully for all functions except for f_2 and f_8. When the execution time is investigated, it is easy to see that there is quite a big difference between algorithms. We primarily attribute this difference to distinct computers used for the simulations, and secondly to the fact that the computational burden of the proposed algorithm is lighter than that of CSOS algorithm.

To highlight the convergence superiority of ISOS algorithm, its convergence curves for f_1, f_4 f_7, and f_{11} are shown in Fig. 2. It is to be noted that since the y-axis data is displayed with logarithmic scale, the last value shown in these convergence profiles points out the final value just before the algorithm finds the theoretically global minimum.

4.1.2. Results for four, five and ten-dimensional problems

The optimization results for four, five and ten-dimensional test functions are reported in Table 6. It is remarkable that ISOS outperforms its competitor for f_{12} and f_{14} by successfully reaching the global

Table 3
The details of benchmark test functions with thirty dimensions.

Function	Range	Type	D	Formulation	Min
Step (f_{16})	[−5.12, 5.12]	US	30	$f_{16}(u)=\sum_{i=1}^{D}\left(u_i+0.5\right)^2$	0
Sphere (f_{17})	[−100, 100]	US	30	$f_{17}(u)=\sum_{i=1}^{D}u_i^2$	0
Sum squares (f_{18})	[−10, 10]	US	30	$f_{18}(u)=\sum_{i=1}^{D}iu_i^2$	0
Quartic (f_{19})	[−1.28, 1.28]	US	30	$f_{19}(u)=\sum_{i=1}^{D}iu_i^4+Rand$	0
Schwefel 2.22 (f_{20})	[−10, 10]	UN	30	$f_{20}(u)=\sum_{i=1}^{D}\lvert u_i\rvert+\prod_{i=1}^{D}\lvert u_i\rvert$	0
Schwefel 1.2 (f_{21})	[−100, 100]	UN	30	$f_{21}(u)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}u_j\right)^2$	0
Rosenbrock (f_{22})	[−30, 30]	MN	30	$f_{22}(u)=\sum_{i=1}^{D-1}100\left(u_{i+1}-u_i^2\right)^2+\left(u_i-1\right)^2$	0
Dixon-Price (f_{23})	[−10, 10]	UN	30	$f_{23}(u)=\left(u_1-1\right)^2+\sum_{i=2}^{D}i\left(2u_i^2-u_i-1\right)^2$	0
Rastrigin (f_{24})	[−5.12, 5.12]	MS	30	$f_{24}(u)=\sum_{i=1}^{D}\left(u_i^2-10\cos\left(2\pi u_i\right)+10\right)$	0
Griewank (f_{25})	[−600, 600]	MN	30	$f_{25}(u)=\frac{1}{4000}\left(\sum_{i=1}^{D}\left(u_i-100\right)^2\right)-\left(\prod_{i=1}^{D}\cos\left(\frac{u_i-100}{\sqrt{i}}\right)\right)+1$	0
Ackley (f_{26})	[−32, 32]	MN	30	$f_{26}(u)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}u_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{D}\cos\left(2\pi u_i\right)\right)+20+e$	0

D: Dimension, U: Unimodal, M: Multimodal, N: non-separable, S: Separable.

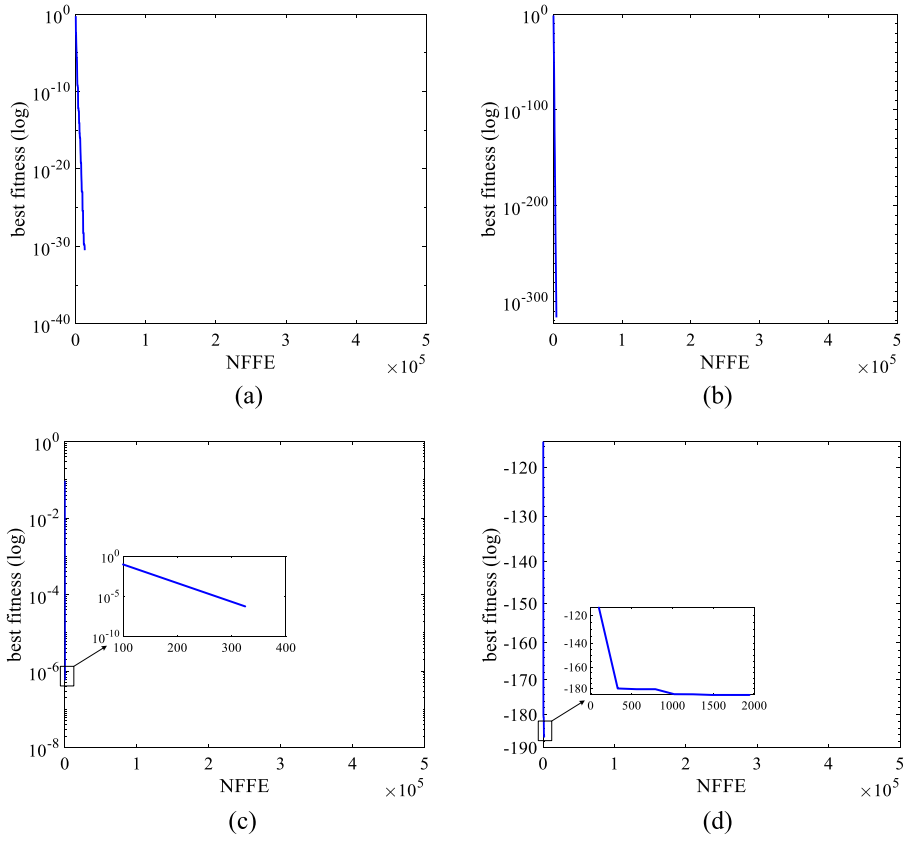

Fig. 2. Convergence trendlines of ISOS on some two-dimensional benchmarks (a) f_1 (b) f_4 (c) f_7 (d) f_{11}.

minima in each run. On the other hand, both algorithms are able to achieve the minimum point of f_{13} and for the other test function, ISOS produces a promising result very closer to the global minimum. When the NFFE and execution time are examined, ISOS is notable as it requires significantly fewer NFFEs and lower execution time in comparison with CSOS excepting f_{14}, in which case ISOS, however, provides the global minimum.

The convergence characteristics of ISOS algorithm showing the best fitness trajectory are demonstrated in Fig. 3 for the functions f_{12} and f_{14}. It is obvious from Fig. 2 that the optimal values of the respective functions are found faster than CSOS in the case study.

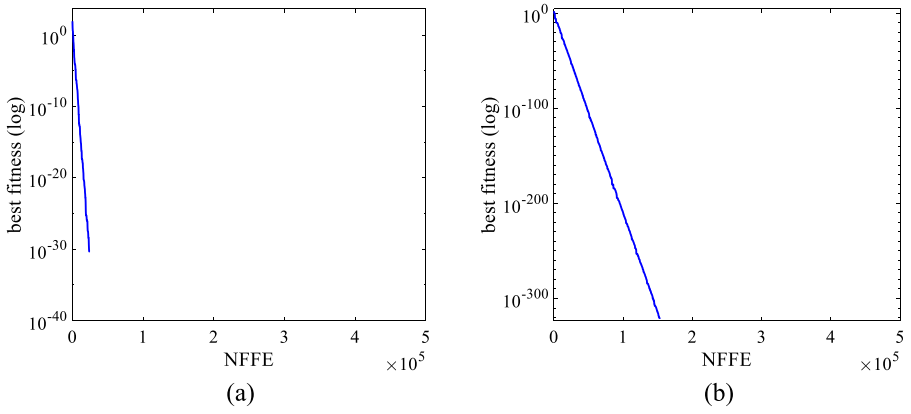

Fig. 3. Convergence trendlines of ISOS on the four and ten-dimensional benchmarks (a) f_{12} (b) f_{14}.

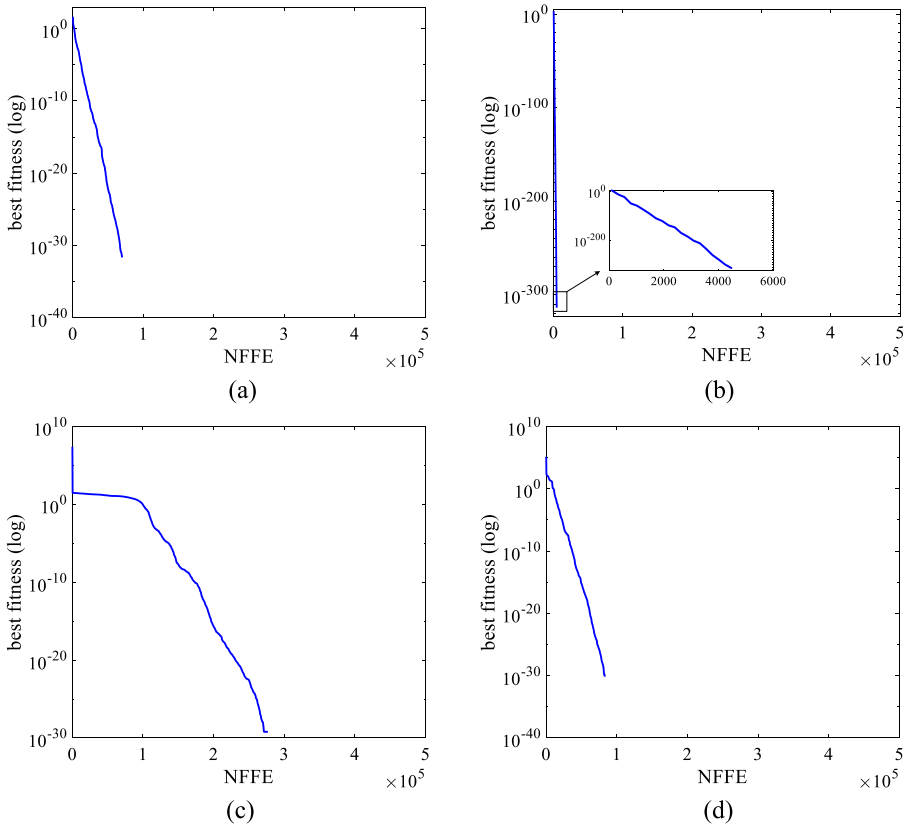

Fig. 4. Convergence trendlines of ISOS on some thirty-dimensional benchmarks (a) f_{16} (b) f_{18} (c) f_{22} (d) f_{23}.

Table 4
The tuning parameters required in different algorithms for global optimization.

Algorithm	Number of parameters	Tuning parameters
CSOS (Saha and Mukherjee, 2018)	5	Ecosystem size $N = 50$, maximum number of fitness function evaluations $NFFE = 500{,}000$, chaotic search radius r, shrinking coefficient δ, iteration number for chaotic search K
ISOS [proposed]	3	Ecosystem size $N = 50$, maximum number of fitness function evaluations $NFFE = 500{,}000$, iteration number for chaotic search $K = 100$

4.1.3. Results for thirty-dimensional problems

Regarding the thirty-dimensional test functions popular in literature, the optimization findings of the proposed ISOS algorithm and previously reported CSOS algorithm are tabulated in Table 7. As signified by bold faces, ISOS yields better mean results than its earlier counterpart for 10 of the 11 functions where the true minimums are effectively detected for f_{16}, f_{17}, f_{18}, f_{20}, f_{21}, f_{23}, and f_{24}. What is more important is that the proposed algorithm offers better solution quality while ensuring significantly fewer NFFEs in most of the test functions. This accordingly brings an improvement in time consumption.

Fig. 4 shows the convergence capability of the ISOS algorithm when minimizing f_{16}, f_{18}, f_{22}, and f_{23}. Notice that Fig. 3(c) visualizes the search process when the proposed ISOS algorithm is able to gain the global optimum value of the extremely difficult Rosenbrock (function f_{22}).

Inspecting the reported values of Tables 5–7 and Figs. 2–4 indicates that the suggested integration scheme is beneficial satisfying both solution accuracy and convergence rate. The reason accounting for such performance enhancement is that the ISOS algorithm induces

Table 5
Comparative results between CSOS and ISOS for two-dimensional problems.

Function	Algorithm	CSOS (Saha and Mukherjee, 2018)	ISOS [Proposed]
f_1	Mean	2.8463E−32	**0**
	StDev	2.2355E−31	**0**
	NFFE	22,200	**14,868**
	Time (s)	10.18	**0.161**
	Rank	2	**1**
f_2	Mean	−1	−1
	StDev	0	0
	NFFE	4275	5530
	Time (s)	10.27	**0.093**
	Rank	1	1
f_3	Mean	4.583E−324	**0**
	StDev	0	**0**
	NFFE	112,560	**5793**
	Time (s)	10.35	**0.096**
	Rank	2	**1**
f_4	Mean	2.2204E−16	**0**
	StDev	5.3469E−17	**0**
	NFFE	2265	**545**
	Time (s)	10.07	**0.048**
	Rank	2	**1**
f_5	Mean	1.2648E−30	**0**
	StDev	2.3516E−30	**0**
	NFFE	22,060	**12,317**
	Time (s)	10.25	**0.099**
	Rank	2	**1**
f_6	Mean	−1.8013	−1.8013
	StDev	0	0
	NFFE	1320	**1292**
	Time (s)	10.23	**0.059**
	Rank	1	1
f_7	Mean	4.4588E−17	**0**
	StDev	1.8166E−18	**0**
	NFFE	6945	**865**
	Time (s)	10.25	**0.058**
	Rank	2	**1**
f_8	Mean	−1.03163	−1.03163
	StDev	0	0
	NFFE	**440**	1000
	Time (s)	10.53	**0.055**
	Rank	1	1
f_9	Mean	6.4525E−17	**0**
	StDev	3.3284E−17	**0**
	NFFE	2850	**475**
	Time (s)	10.16	**0.049**
	Rank	2	**1**
f_{10}	Mean	4.5751E−17	**0**
	StDev	2.4218E−17	**0**
	NFFE	3150	**520**
	Time (s)	10.05	**0.054**
	Rank	2	**1**
f_{11}	Mean	−186.73	−186.73
	StDev	0	0
	NFFE	354,600	**2216**
	Time (s)	10.79	**0.071**
	Rank	1	1
Average rank		1.6363	**1**
Overall rank		2	**1**

exploration in the initial iterations of optimization and then gradually transits to exploitation in the last iterations of optimization by making a direct search around promising regions nearby the best solution. This significantly assists this algorithm to speed up the search speed. Fast convergence speed becomes an important necessity in online optimization applications where the optimal values should be quickly found and updated in the process within two sampling intervals.

Table 6
Comparative results between CSOS and ISOS for four, five and ten-dimensional problems.

Function	Algorithm	CSOS (Saha and Mukherjee, 2018)	ISOS [Proposed]
f_{12}	Mean	1.0741E−25	**0**
	StDev	1.1253E−24	**0**
	NFFE	499,950	**25,750**
	Time (s)	12.55	**0.243**
	Rank	2	**1**
f_{13}	Mean	−4.6877	−4.6877
	StDev	0	0
	NFFE	9300	**4045**
	Time (s)	11.03	**0.092**
	Rank	1	**1**
f_{14}	Mean	2.125E−323	**0**
	StDev	0	0
	NFFE	**120,000**	155,690
	Time (s)	11.27	**1.350**
	Rank	2	**1**
f_{15}	Mean	**−9.6602**	−9.6505
	StDev	**0**	0.019
	NFFE	495,600	**105,460**
	Time (s)	12.85	**0.907**
	Rank	1	2
Average rank		1.5	**1.2500**
Overall rank		2	**1**

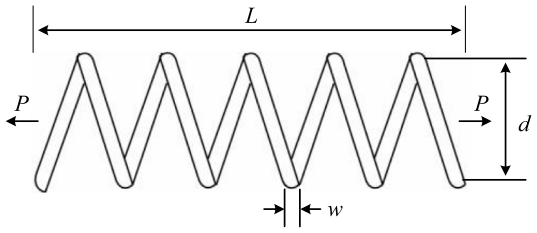

Fig. 5. Visualization of tension/compression spring design problem.

4.2. Engineering design problems

So far, the searching capability of ISOS algorithm is explored on the unconstrained benchmark functions of different kinds and dimensions. In order to further testify the acceptability of proposed approach in presence of constrained global optimization problems, three real-world engineering design problems are taken from the literature and solved using the presented ISOS algorithm. The selected problems, which are tension/compression spring design, pressure vessel design and PID controlled automatic voltage regulator (AVR) design, have been already resolved by many other stochastic techniques available in literature. The obtained results are compared to the earlier ones reported by the addressed optimization techniques for each problem to betoken the eminence of proposed algorithm. In any of the reported tables giving the respective statistical data based on the proposed ISOS algorithm, 20 decimal places are considered in the calculation of standard deviation.

4.2.1. Tension/compression spring design problem

This problem shown in Fig. 5 corresponds to minimization of the weight of a tension/compression spring subject to a set of constraints on such as shear stress, surge frequency and minimum deflection.

The design optimization problem consists of three continuous variables and four nonlinear inequality constraints. Its mathematical formulation is expressed as:

$$\text{Minimize } f(w,d,L) = (L+2)\,d w^2 \tag{15}$$

Subject to

Table 7
Comparative results between CSOS and ISOS for thirty-dimensional problems.

Function	Algorithm	CSOS (Saha and Mukherjee, 2018)	ISOS [Proposed]
f_{16}	Mean	2.2244E−35	**0**
	StDev	1.1112E−34	**0**
	NFFE	**78,500**	83,867
	Time (s)	11.20	**0.837**
	Rank	2	**1**
f_{17}	Mean	1.443E−322	**0**
	StDev	0	0
	NFFE	89,700	**4930**
	Time (s)	11.95	**0.104**
	Rank	2	**1**
f_{18}	Mean	1.538E−320	0
	StDev	0	0
	NFFE	88,650	**4930**
	Time (s)	11.56	**0.109**
	Rank	2	**1**
f_{19}	Mean	2.4804E−05	**3.3979E−06**
	StDev	2.2144E−06	**2.0697E−06**
	NFFE	500,000	500,000
	Time (s)	12.15	**6.245**
	Rank	2	**1**
f_{20}	Mean	1.242E−162	**0**
	StDev	0	0
	NFFE	176,700	**9086**
	Time (s)	12.39	**0.154**
	Rank	2	**1**
f_{21}	Mean	3.593E−321	**0**
	StDev	0	0
	NFFE	90,450	**5076**
	Time (s)	11.50	**0.110**
	Rank	2	**1**
f_{22}	Mean	0.4208	**2.5185E−29**
	StDev	0.8653	**1.5790E-29**
	NFFE	500,000	**489,530**
	Time (s)	11.52	**4.210**
	Rank	2	**1**
f_{23}	Mean	0.667	**0**
	StDev	0	0
	NFFE	500,000	**83,087**
	Time (s)	11.24	**0.812**
	Rank	2	**1**
f_{24}	Mean	3.1221E−14	**0**
	StDev	1.2551E−14	**0**
	NFFE	11,000	**550**
	Time (s)	11.34	**0.051**
	Rank	2	**1**
f_{25}	Mean	**1.1016E−16**	4.1089E−04
	StDev	**1.1338E−16**	1.7433E−03
	NFFE	**10,660**	500,000
	Time (s)	11.12	**4.213**
	Rank	**1**	2
f_{26}	Mean	3.7224E−15	**8.8817E−16**
	StDev	1.1268E−15	**0**
	NFFE	500,000	500,000
	Time (s)	12.35	**4.883**
	Rank	2	**1**
Average rank		1.9090	**1.0909**
Overall rank		2	**1**

$$\begin{cases} g_1 = 1 - \dfrac{d^3 L}{71785 w^3} \le 0 \\ g_2 = \dfrac{d\,(4d - w)}{12566 w^3\,(d - w)} + \dfrac{1}{5108 w^2} - 1 \le 0 \\ g_3 = 1 - \dfrac{140.45 w}{d^2 L} \le 0 \\ g_4 = \dfrac{w + d}{1.5} - 1 \le 0 \end{cases} \tag{16}$$

where w, d and L are wire diameter, mean coil diameter and the number of spring's active coils (or its length), respectively. The upper and lower limits of these design variables are:

$$0.05 \le w \le 2, \qquad 0.25 \le d \le 1.3, \qquad 2.0 \le L \le 15.0 \tag{17}$$

As stated before, there have been existing many results in literature regarding this problem reported based on several approaches like coevolutionary particle swarm optimization (CEPSO) (Krohling and Coelho, 2006), coevolutionary differential evolution (CEDE) (Huang et al., 2007), water cycle algorithm WCA (Eskandar et al., 2012), grey wolf optimizer (Mirjalili et al., 2014), quasi-oppositional chaotic symbiotic organisms search (QOCSOS) (Truong et al., 2019), and stochastic fractal search (SFS) (Salimi, 2015). The best results achieved by the present work are compared to six recent solutions, and are shown in Table 8. It is obvious from this table that both SFS and ISOS yield the same weight value which is superior to that of the other five competitors.

The respective statistical performance of ISOS and other cited algorithms after 30 independent runs is gathered in Table 9. By dint of this table, it becomes clear that ISOS outperforms its closest competitor SFS as it can gain a zero standard deviation value and converge towards the minimum weight value in 2.5 times fewer NFFEs than SFS technique.

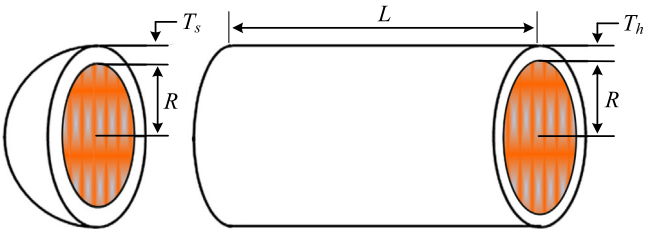

Fig. 6. Visualization of pressure vessel design problem.

4.2.2. Pressure vessel design problem

Pressure vessels are an inseparable part of several manufacturing facilities and processing plants, providing reliable storage of pressurized liquids and gases. A pressure vessel is commonly designed using a cylinder enclosed with end caps/heads that are usually hemispherical as illustrated in Fig. 6. The goal of this problem is to minimize the total cost of fabricating this structure including material, forming and welding costs.

The problem includes four design variables to be optimized:

- Thickness of the shell (T_s)
- Thickness of the head (T_h)
- Inner radius (R)
- Length of the cylindrical section of the vessel (L)

Among these variables, T_s and T_h are discrete values in the order of integer multiplies of 0.0625, and R and L are continuous values, the lower and upper bounds of which are given below.

$$1 \times 0.0625 \le T_s, \qquad T_h \le 99 \times 0.0625, \qquad 10.0 \le R \le 200.0$$
$$10.0 \le L \le 200.0 \tag{18}$$

The formulation of this problem along with the four inequality constraints can be mathematically given as follows:

$$\text{Minimize } f\left(T_s, T_h, R, L\right) = 0.6224 T_s R L + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_s^2 R \tag{19}$$

Subject to

$$\begin{cases} g_1 = -T_s + 0.0193 R \le 0 \\ g_2 = -T_h + 0.0095 R \le 0 \\ g_3 = -\pi R^2 L - \dfrac{4}{3}\pi R^3 + 1296000 \le 0 \\ g_4 = L - 240 \le 0 \end{cases} \tag{20}$$

Table 8
Comparison of the best results for tension/compression spring design problem offered by different techniques.

	CEPSO (Krohling and Coelho, 2006)	CEDE (Huang et al., 2007)	WCA (Eskandar et al., 2012)	GWO (Mirjalili et al., 2014)	QOCSOS (Truong et al., 2019)	SFS (Salimi, 2015)	ISOS [proposed]
w	0.0517280	0.0516090	0.051689	0.051690	0.055130816	0.051689060916152	0.051689061903120
d	0.3576440	0.3547140	0.356522	0.356737	0.444775663	0.356717735791209	0.356717759535058
L	11.244543	11.410831	11.30041	11.28885	7.544716912	11.288965986603480	11.288964594575669
f	0.0126740	0.0126702	0.012665	0.012666	0.012903065	**0.012665232788319**	**0.012665232788319**

Table 9
Comparing the statistical performance of ISOS and other algorithms for tension/compression spring design problem.

	CEPSO (Krohling and Coelho, 2006)	CEDE (Huang et al., 2007)	WCA (Eskandar et al., 2012)	GWO (Mirjalili et al., 2014)	QOCSOS (Truong et al., 2019)	SFS (Salimi, 2015)	ISOS [proposed]
Best	0.012674	0.0126702	0.012665	0.012666	0.012903065	**0.012665232788319**	**0.012665232788319**
Mean	0.012730	0.0126703	0.012746	NA	NA	**0.012665232788319**	**0.012665232788319**
Worst	0.012924	0.0126790	0.012952	NA	NA	**0.012665232788319**	**0.012665232788319**
StDev	1.58E−05	2.70E−05	8.06E−06	NA	NA	1.5858E−16	**0**
NFFE	240,000	204,800	11,750	NA	NA	100,000	**40,000**

Similar to the previous problem, pressure vessel design problem has been popularly used as a benchmark by the researches and solved in several studies. CEPSO (Krohling and Coelho, 2006), comprehensive learning PSO algorithm (Gao and Hailu, 2010), mine blast algorithm (MBA) (Sadollah et al., 2013), QOCSOS (Truong et al., 2019), GWO (Mirjalili et al., 2014) and SFS (Salimi, 2015) are some among the recent studies of interest, and their best solutions along with the value of f are listed in Table 10 in comparison with that given by the proposed algorithm. It may be observed from this table that both SFS and ISOS yield again similar performance in achieving a design with the most promising cost. It is worth signifying that although the f values acquired by MBA, GWO and QOCSOS seem smaller than the others, they are not feasible because the design variables T_s and/or T_h are not discrete in the integer multiplies of 0.0625. On the other hand, the cost function value derived by PSO is comparable with ISOS, but the suggested value is infeasible since the reported solution violates the constraint g_3.

The respective statistical performance of ISOS and above indicated algorithms for the case study is tabulated in Table 11. As seen, SFS and ISOS are the most reliable and robust algorithms comparing to the earlier studies, ensuring always the same best, mean and worst solutions. However, although SFS matched the performance of ISOS with regard to the best, mean and worst performance, ISOS is still the only pioneer in terms of standard deviation value equal to zero and fewer NFFEs. As such, we can conclude that the proposed algorithm of the current study is the most effective algorithm in optimizing the design variables of a pressure vessel.

The following subsection judges the performance of the ISOS algorithm in answering to the design concern of a PID controlled automatic voltage regulator (AVR) system in the hot field of electric power system.

4.2.3. PID controlled AVR design problem

An AVR is a system that regulates the terminal voltage of a synchronous generator at a nominal constant voltage level by controlling the machine excitation current. In addition to its inherent cost advantage, excitation control of a synchronous alternator is one of the most significant factors to enhance power system stability and security as well as the quality of produced electrical power (Shayeghi et al., 2015). A basic AVR model combines five sub-models of sensor, amplifier, exciter, and generator. The schematic representation of this AVR system and its transfer function block diagram including PID controller can be given as in Fig. 7.

The selected parameters for the AVR system are publicly available in literature, which are, respectively, $K_a = 10$, $\tau_a = 0.1$, $K_e = 1$, $\tau_e = 0.4$, $K_g = 1$, $\tau_g = 1$, $K_s = 1$ and $\tau_s = 0.01$ (Pandaa et al., 2012; Mohanty et al., 2014; Güvenç et al., 2016; Çelik, 2018). The objective of this design problem is to tune proportional gain (K_p), integral gain (K_i) and derivative gain (K_d) of the PID controller so that the terminal voltage profile of the AVR system may be the most preferable and desirable, which is achieved when the values of rise time, settling time, overshoot, and peak time associated with the unit step input are minimum. The following performance criterion is assumed as objective function in line with the existing research works in order to minimize the said time-domain performance parameters of the terminal voltage.

$$f\left(K_p, K_i, K_d\right) = ITSE = \int_0^{t_{sim}} t(\Delta e)^2 dt \tag{21}$$

where f is also known as integral of time square error (ITSE), Δe is voltage error equal to $\Delta e = \left(\Delta V_{ref} - \Delta V_s\right)$ and t_{sim} is simulation run time.

The constraints of the present optimization problem are the gains of the PID controller, which must be bounded within certain limits. That said, tuning concern of PID gains is defined by the following formulation:

$$\text{Minimize } f \tag{22}$$

Subject to

$$\begin{cases} K_p^{min} \le K_p \le K_p^{max} \\ K_i^{min} \le K_i \le K_i^{max} \\ K_d^{min} \le K_d \le K_d^{max} \end{cases} \tag{23}$$

where the superscripts *min* and *max* stand for the minimum and maximum bounds of the respective controller parameter. Depending on the detailed literature inspection, all the gains are similarly assumed in the range [0.2, 2.0].

This problem is solved by a variety of optimization tools like many optimizing liaisons (MOL) algorithm (Pandaa et al., 2012), local unimodal sampling (LUS) optimization algorithm (Mohanty et al., 2014), biogeography-based optimization (BBO) (Güvenç et al., 2016), and SFS (Çelik, 2018). The remainder of this section applies ISOS algorithm to solve the considered problem and compares its solutions to solutions reported by the indicated powerful algorithms under identical conditions. The controller gains tuned by using ITSE objective function as well as their performance are provided in Table 12 for various algorithms. It is evident from this table that the introduced ISOS based PID controller validates its contribution in achieving the minimum ITSE value of the AVR terminal voltage compared to other existing approaches.

A comparison of the convergence characteristics between SFS and ISOS algorithm can be viewed from Fig. 8. For a fair comparison, the algorithms are initiated with the same population at iteration zero. As clearly seen, ISOS has also faster convergence rate than SFS. This comparison elucidates the fact that the proposed ISOS algorithm contributes to both convergence speed and solution accuracy simultaneously in the studied case study.

Table 10
Comparison of the best results for pressure vessel design problem offered by different techniques.

	CEPSO (Krohling and Coelho, 2006)	PSO (Gao and Hailu, 2010)	MBA (Sadollah et al., 2013)	GWO (Mirjalili et al., 2014)	QOCSOS (Truong et al., 2019)	SFS (Salimi, 2015)	ISOS [proposed]
T_s	0.8125	0.8125	0.7802	0.8125	0.778238	0.8125	0.8125 (13×0.0625)
T_h	0.4375	0.4375	0.3856	0.4345	0.384893	0.4375	0.4375 (7×0.0625)
R	42.0913	42.0984	40.4292	42.089181	40.322081	42.09844559585492	42.09844559585492
L	176.7465	176.6366	198.4964	176.758731	199.966711	176.6365958424395	176.6365958424395
f	6061.0777	6059.7143	5889.3216	6051.5639	5885.332774	**6059.714335048436**	**6059.714335048436**

Table 11
Comparing the statistical performance of ISOS and other algorithms for pressure vessel design problem.

	CEPSO (Krohling and Coelho, 2006)	PSO (Gao and Hailu, 2010)	MBA (Sadollah et al., 2013)	GWO (Mirjalili et al., 2014)	QOCSOS (Truong et al., 2019)	SFS (Salimi, 2015)	ISOS [proposed]
Best	6061.0777	6059.7143	5889.3216	6051.5639	5885.332774	**6059.714335048436**	**6059.714335048436**
Mean	6147.1332	6066.0311	6200.64765	NA	NA	**6059.714335048436**	**6059.714335048436**
Worst	6363.8041	NA	6392.5062	NA	NA	**6059.714335048436**	**6059.714335048436**
StDev	86.4500	12.2718	160.34	NA	NA	9.5869E−13	**0**
NFFE	240,000	60,000	70,650	NA	NA	50,000	**15,000**

Table 12
Optimized parameters of PID controller for AVR system using different algorithms.

	MOL (Pandaa et al., 2012)	LUS (Mohanty et al., 2014)	BBO (Güvenç et al., 2016)	SFS (Çelik, 2018)	ISOS [proposed]
K_p	0.9877	1.2012	1.2464	1.283695289285423	1.283678042285351
K_i	0.7780	0.9096	0.5893	1.339299310920850	1.339229429513187
K_d	0.5014	0.4593	0.4596	0.777988728439710	0.777964377983033
ITSE	0.0062	0.0064	0.0073	0.005266089999403	**0.005266089993638**

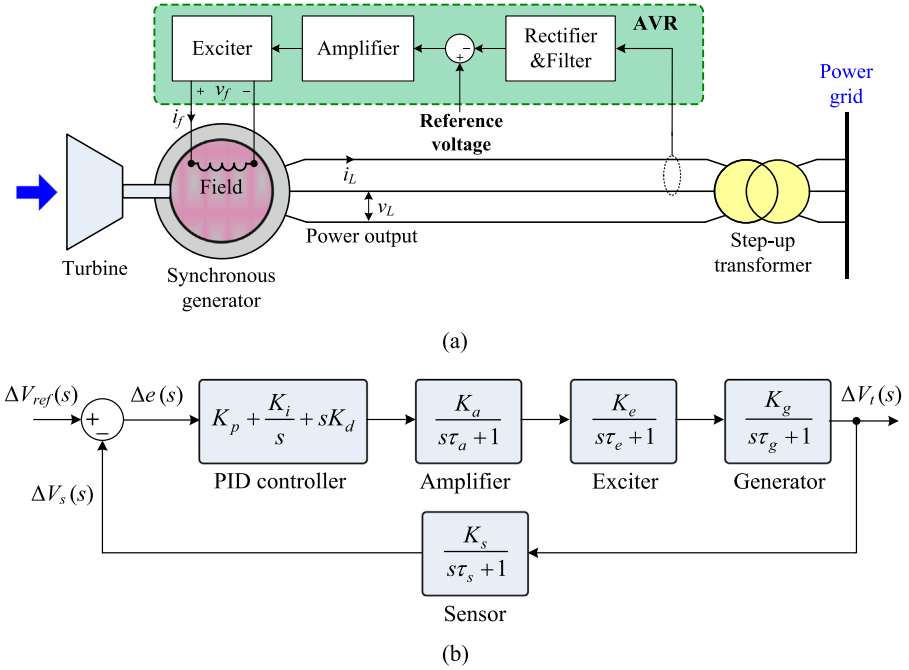

Fig. 7. AVR system (a) schematic representation (b) transfer function block diagram.

5. Conclusions

This article has introduced a powerful variant of SOS algorithm named improved SOS (ISOS). Inspiration for the proposed method arises because of the fact that the original SOS has difficulty in balancing between exploration and exploitation capabilities, and also its parasitism phase results in over exploration that significantly diminishes the computational efficiency of the algorithm. ISOS makes use of QOBL after generating the initial random population and in the parasitism phase to obtain high-quality solutions quickly. Another effective strategy is also introduced for parasitism phase. To further improve the solution quality and convergence speed of ISOS, PWLCM based CLS is integrated into the proposed algorithm to intensify the search process around the global best point where the chaotic search range decreases naturally as the solutions converge. Thanks to these modifications, ISOS performance is greatly boosted compared to its original version and many other evolutionary algorithms used. To verify this, 26 test functions with different types and dimensions are adopted to benchmark the performance of the presented algorithm. The efficacy of ISOS is also tested with three engineering design problems popular in the field of interest. All the results are carefully evaluated in terms of solution quality and convergence mobility. The numerical results are surprising and affirm that in quite most cases ISOS is

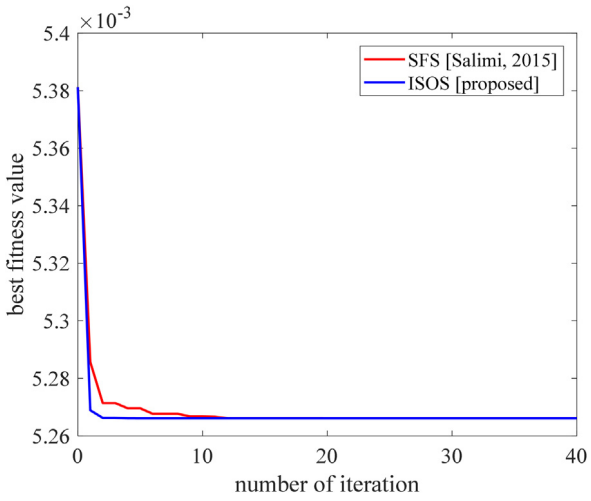

Fig. 8. Convergence rates for the PID controlled AVR system using SFS and ISOS algorithms.

found to offer more accurate results with far fewer function evaluations than algorithms tested in earlier works. This is attributed to the effectiveness of the operators that are developed to allow ISOS to converge towards the optimum point quickly by successfully avoiding local optima stagnation, which may become important particularly for online optimization challenges. In addition, the proposed algorithm provides the said merits by only requiring one tuning parameter. This increases the algorithm robustness and accordingly reduces the possibility of inefficient search performance owing to incorrect parameter tuning. It has been shown that ISOS could contribute to answering the solutions of various benchmark functions and some practical problems considered in this paper. Evaluating its performance on other areas in need of optimization is vital in line with the "no free lunch" theorem, and can thereby be contemplated as future study. Moreover, inclusion of other chaotic maps within search process is also worthy of attention.

References

Alatas, B., Akin, E., Ozer, A.B., 2009. Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals 40 (4), 1715–1734.

Çelik, E., 2018. Incorporation of stochastic fractal search algorithm into efficient design of PID controller for an automatic voltage regulator system. Neural Comput. Appl. 30 (6), 1991–2002.

Çelik, E., Durgut, R., 2018. Performance enhancement of automatic voltage regulator by modified cost function and symbiotic organisms search algorithm. Eng. Sci. Technol. Int. J. 21 (5), 1104–1111.

Çelik, E., Öztürk, N., 2018a. First application of symbiotic organisms search algorithm to off-line optimization of PI parameters for DSP-based DC motor drives. Neural Comput. Appl. 30 (5), 1689–1699.

Çelik, E., Öztürk, N., 2018b. A hybrid symbiotic organisms search and simulated annealing technique applied to efficient design of PID controller for automatic voltage regulator. Soft Comput. 22 (23), 8011–8024.

Cheng, M.Y., Prayogo, D., 2014. Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput. Struct. 139, 98–112.

Coelho, L.S., Mariani, V.C., 2012. Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Comput. Math. Appl. 64 (8), 2371–2382.

Eskandar, H., Sadollah, A., Bahreininejad, A., Hamdi, M., 2012. Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput. Struct. 110–111, 151–166.

Gandomi, A., Yang, X.S., Talatahari, S., Alavi, A., 2012. Firefly algorithm with chaos. Commun. Nonlinear Sci. 18 (1), 89–98.

Gao, L., Hailu, A., 2010. Comprehensive learning particle swarm optimizer for constrained mixed-variable optimization problems. Int. J. Comput. Int. Syst. 3 (6), 832–842.

Guhaa, D., Roy, P., Banerjee, S., 2017. Quasi-oppositional symbiotic organism search algorithm applied to load frequency control. Swarm Evol. Comput. 33, 46–67.

Güvenç, U., Yazıcı, A., Yılmaz, C., 2018. Dynamic economic dispatch using chaos based gravitational search algorithm. In: 7th International Conference on Advanced Technologies, 28 April-1 Antalya, Turkey.

Güvenç, U., Yiğit, T., Işık, A.H., Akkaya, İ., 2016. Performance analysis of biogeography-based optimization for automatic voltage regulator system. Turk. J. Electr. Eng. Comput. Sci. 24, 1150–1162.

Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267, 66–72.

Huang, F.Z., Wang, L., He, Q., 2007. An effective co-evolutionary differential evolution for constrained optimization. Appl. Math. Comput. 186 (1), 340–356.

Jain, Mohit, Singh, Vijander, Rani, Asha, 2019. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm Evol. Comput. 44, 148–175.

Kaplan, O., Çelik, E., 2018. Simplified model and genetic algorithm based simulated annealing approach for excitation current estimation of synchronous motor. Adv. Electr. Comput. Enf. 18 (4), 75–84.

Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. 39, 459–471.

Karkheiran, S., Samani, A.K., Zekri, M., Azamathulla, H.M., 2019. Scour at bridge piers in uniform and armored beds under steady and unsteady flow conditions using ANN-APSO and ANN-GA algorithms. ISH J. Hydraul. Eng. http://dx.doi.org/10.1080/09715010.2019.1617796.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: IEEE International Conference on Neural Networks, 27 Nov.-1 Dec. Perth, WA, Australia, Australia.

Kim, D.H., Abraham, A., Cho, J.H., 2007. A hybrid genetic algorithm and bacterial foraging approach for global optimization. Inform. Sci. 177, 3918–3937.

Krohling, R.A., Coelho, L.S., 2006. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. IEEE Trans. Syst. Man Cybern. B 36 (6), 1407–1416.

Mirjalili, S., 2015. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl. Based Syst. 89, 228–249.

Mirjalili, S., Gandomi, A.H., 2017. Chaotic gravitational constants for the gravitational search algorithm. Appl. Soft Comput. 53, 407–419.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.

Mohanty, P.K., Sahu, B.K., Panda, S., 2014. Tuning and assessment of proportional–integral–derivative controller for an automatic voltage regulator system employing local unimodal sampling algorithm. Electr. Power Comput. Syst. 42, 959–969.

Padhy, S., Panda, S., 2017. A hybrid stochastic fractal search and pattern search technique based cascade PI-PD controller for automatic generation control of multi-source power systems in presence of plug in electric vehicles. CAAI Trans. Intell. Technol. 2, 12–25.

Pandaa, S., Sahu, B.K., Mohanty, P.K., 2012. Design and performance analysis of PID controller for an automatic voltage regulator system using simplified particle swarm optimization. J. Frankl. Inst. 349 (8), 2609–2625.

Pistolesi, F., Lazzerini, B., Mura, M.D., Dini, G., 2018. EMOGA: A hybrid genetic algorithm with extremal optimization core for multiobjective disassembly line balancing. IEEE Trans. Ind. Inform. 14 (3), 1089–1098.

Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A., 2007. Quasi oppositional differential evolution. In: IEEE congress on Evolutionary Computation, 25-28 Sept. Singapore, Singapore.

Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A., 2008a. Opposition-based differential evolution. IEEE Trans. Evol. Comput. 12 (1), 64–79.

Rahnamayan, S., Tizhoosh, H.R., Salma, M.M.A., 2008b. Opposition versus randomness in soft computing technique. Appl. Soft Comput. 8 (2), 906–918.

Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S., 2009. GSA: a gravitational search algorithm. Inf. Sci. 179, 2232–2248.

Roy, P.K., Bhui, S., 2013. Multi-objective quasi-oppositional teaching learning based optimization for economic emission load dispatch problem. Int. J. Electr. Power 53, 937–948.

Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M., 2013. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl. Soft Comput. 13 (5), 2592–2612.

Saha, A., Chakraborty, A.K., Das, P., 2019. Quasi-reflection based symbiotic organisms search algorithm for solving static optimal power flow problem. Sci. Iran. 26 (3), 1664–1689.

Saha, S., Mukherjee, V., 2016. Optimal placement and sizing of DGs in RDS using chaos embedded SOS algorithm. IET Gener. Transm. Distrib. 10 (14), 3671–3680.

Saha, S., Mukherjee, V., 2018. A novel chaos-integrated symbiotic organisms search algorithm for global optimization. Soft Comput. 22, 3797–3816.

Salimi, H., 2015. Stochastic fractal search: a powerful metaheuristic algorithm. Knowl. Based Syst. 75, 1–18.

Saremi, S., Mirjalili, S.M., Mirjalili, S., 2014. Chaotic krill herd optimization algorithm. Procedia Technol. 12, 180–185.

Sedighizadeh, M., Esmailib, M., Eisapour-Moarref, A., 2017. Voltage and frequency regulation in autonomous microgrids using Hybrid Big Bang-Big Crunch algorithm. Appl. Soft Comput. 52, 176–189.

Shayeghi, H., Younesi, A., Hashemi, Y., 2015. Optimal design of a robust discrete parallel FP + FI + FD controller for the Automatic Voltage Regulator system. Int. J. Electr. Power 67, 66–75.

Shiva, C.K., Shankar, G., Mukherjee, V., 2015. Automatic generation control of power system using a novel quasi-oppositional harmony search algorithm. Int. J. Electr. Power 73, 787–804.

Tejani, G.G., Savsanin, V.J., Patel, V.K., 2016. Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization. J. Comput. Des. Eng. 3 (3), 226–249.

Tizhoosh, H.R., 2005. Opposition-based learning: a new scheme for machine intelligence. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, 28-30 Nov. Vienna, Austria.

Truong, K.H., Nallagownden, P., Baharudin, Z., Vo, D.N., 2019. A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems. Appl. Soft Comput. 77, 567–583.

Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1, 67–82.

Wu, G., Qiu, D., Yu, Y., Pedrycz, W., Ma, M., Li, H., 2014. Superior solution guided particle swarm optimization combined with local search techniques. Expert Syst. Appl. 41, 7536–7548.

Xiang, T., Liao, X., Wong, K.W., 2007. An improved particle swarm optimization algorithm combined with piecewise linear chaotic map. Appl. Math. Comput. 190, 1637–1645.

Yu, V.F., Perwira Redi, A.A.N., Yang, C.L., Ruskartina, E., Santosa, B., 2017. Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem. Appl. Soft Comput. 52, 657–672.

Zhang, Zhuoran, Huang, Changqiang, Dong, Kangsheng, Huang, Hanqiao, 2019. Birds foraging search: a novel population-based algorithm for global optimization. Memet Comput. http://dx.doi.org/10.1007/s12293-019-00286-1.