Computers and Structures 139 (2014) 98–112

Contents lists available at ScienceDirect

Computers and Structures

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

Symbiotic Organisms Search: A new metaheuristic optimization algorithm

Min-Yuan Cheng [1], Doddy Prayogo *

Dept. of Civil and Construction Engineering, National Taiwan University of Science and Technology, #43, Sec. 4, Keelung Rd., Taipei 106, Taiwan, ROC

ARTICLE INFO

Article history:
Received 23 May 2013
Accepted 24 March 2014
Available online 10 May 2014

Keywords:
Metaheuristic
Optimization
Symbiotic Organisms Search
Symbiotic interaction
Structural design problems

ABSTRACT

This paper applies a new robust and powerful metaheuristic algorithm called Symbiotic Organisms Search (SOS) to numerical optimization and engineering design problems. SOS simulates the symbiotic interaction strategies adopted by organisms to survive and propagate in the ecosystem. Twenty-six unconstrained mathematical problems and four structural engineering design problems are tested and obtained results compared with other well-known optimization methods. Obtained results confirm the excellent performance of the SOS method in solving various complex numerical problems.

1. Introduction

Engineering optimization is a challenging area of study that has attracted increasing attention in recent decades. Various gradient-based optimization methods have been developed to solve various engineering optimization problems. Most use analytical or numerical methods that require gradient information to improve initial solutions. However, gradient-based optimization methods are inadequate to resolve the complexities inherent in many of today's real-world engineering design problems. Moreover, gradient search in problems with greater than one local optimum is difficult and unstable [1]. Shortcomings in current gradient-based approaches to engineering optimization have thus encouraged researchers to develop better engineering optimization methods.

Research worldwide in the metaheuristic field has produced optimization methods that have proven superior to traditional gradient-based approaches. Osman and Laporte defined metaheuristic as an iterative generation process that integrates different concepts for exploring and exploiting the search space to guide a subordinate heuristic, with learning strategies used to structure information to find efficiently near-optimal solutions [2]. Examples of metaheuristic algorithms include: Genetic Algorithm (GA) [3], Particle Swarm Optimization (PSO) [4], Differential Evolution (DE) [5], Ant Colony Optimization (ACO) [6], Harmony Search (HS) [7], Artificial Bee Colony (ABC) [8], Bees Algorithm (BA) [9], Firefly Algorithm [10], Charge System Search (CSS) [11,12], Big Bang–Big Crunch (BB–BC) [13], Cuckoo Search (CS) [14], Mine Blast Algorithm (MBA) [15], Water Cycle Algorithm [16], Dolphin Echolocation [17], and Ray Optimization [18].

Nearly all metaheuristic algorithms share the same following characteristics: they are nature-inspired; they make use of random variables; they do not require substantial gradient information; and they have several parameters that need to be fitted to the problem at hand [19]. Each metaheuristic algorithm has unique advantages with respect to robustness and performance in noisy environments, in the presence of uncertain parameters, and in different problem spaces [20]. However, in line with the "no-free-lunch" theorem, it is impossible for one metaheuristic algorithm to optimally solve all optimizing problems [21]. Thus, new high-performance metaheuristic algorithms are continuously needed to handle specific optimizing problems.

This paper introduces a new simple and powerful metaheuristic algorithm called Symbiotic Organisms Search (SOS). This algorithm simulates symbiotic interaction strategies that organisms use to survive in the ecosystem. A main advantage of the SOS algorithm over most other metaheuristic algorithms is that algorithm operations require no specific algorithm parameters.

The rest of the paper is organized as follows: Section 2 introduces the SOS algorithm in detail; Section 3 compares SOS performance against well-known algorithms, including GA, DE, PSO, BA, MBA, and CS; Section 4 discusses SOS performance characteristics; and Section 5 presents conclusions.

* Corresponding author at: Dept. of Civil and Construction Engineering, National Taiwan University of Science and Technology, #43, Sec. 4, Keelung Rd., Taipei 106, Taiwan, ROC. Tel.: +886 981892384; fax: +886 2 27301074.

E-mail addresses: myc@mail.ntust.edu.tw (M.-Y. Cheng), doddyprayogo@ymail.com (D. Prayogo).

[1] Tel.: +886 2 27336596; fax: +886 2 27301074.

http://dx.doi.org/10.1016/j.compstruc.2014.03.007
0045-7949/

2. The Symbiosis Organisms Search (SOS) algorithm

The proposed SOS algorithm simulates the interactive behavior seen among organisms in nature. Organisms rarely live in isolation due to reliance on other species for sustenance and even survival. This reliance-based relationship is known as symbiosis. The following subsection clarifies the meaning of symbiosis, gives examples of symbiotic relationship archetypes, and describes the role of symbiosis in ecosystems.

2.1. The basic concept of symbiosis

Symbiosis is derived from the Greek word for "living together". De Bary first used the term in 1878 to describe the cohabitation behavior of unlike organisms [22]. Today, symbiosis is used to describe a relationship between any two distinct species. Symbiotic relationships may be either obligate, meaning the two organisms depend on each other for survival, or facultative, meaning the two organisms choose to cohabitate in a mutually beneficial but nonessential relationship.

The most common symbiotic relationships found in nature are mutualism, commensalism, and parasitism. Mutualism denotes a symbiotic relationship between two different species in which both benefit. Commensalism is a symbiotic relationship between two different species in which one benefits and the other is unaffected or neutral. Parasitism is a symbiotic relationship between two different species in which one benefits and the other is actively harmed.

Fig. 1 illustrates a group of symbiotic organisms living together in an ecosystem. Generally speaking, organisms develop symbiotic relationships as a strategy to adapt to changes in their environment. Symbiotic relationships may also help organisms increase fitness and survival advantage over the long-term. Therefore, it is reasonable to conclude that symbiosis has built and continues to shape and sustain all modern ecosystems.

2.2. The Symbiotic Organisms Search (SOS) algorithm

Current metaheuristic algorithms imitate natural phenomena. For example, Artificial Bee Colony (ABC) simulates the foraging behavior of honeybee swarms, Particle Swarm Optimization simulates animal flocking behavior, and the Genetic Algorithm simulates the process of natural evolution. SOS simulates the symbiotic interactions within a paired organism relationship that are used to search for the fittest organism. The proposed algorithm was developed initially to solve numerical optimization over a continuous search space.

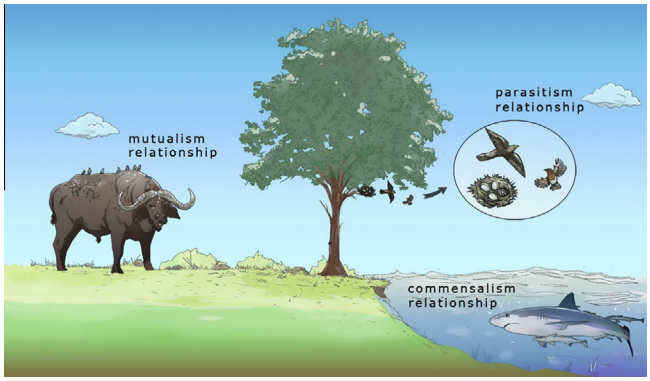

Fig. 1. Symbiotic organisms live together in an ecosystem.

Similar to other population-based algorithms, the proposed SOS iteratively uses a population of candidate solutions to promising areas in the search space in the process of seeking the optimal global solution. SOS begins with an initial population called the ecosystem. In the initial ecosystem, a group of organisms is generated randomly to the search space. Each organism represents one candidate solution to the corresponding problem. Each organism in the ecosystem is associated with a certain fitness value, which reflects degree of adaptation to the desired objective.

Almost all metaheuristic algorithms apply a succession of operations to solutions in each iteration in order to generate new solutions for the next iteration. A standard GA has two operators, namely crossover and mutation. Harmony Search proposes three rules to improvise a new harmony: memory considering, pitch adjusting, and random choosing. Three phases were introduced in the ABC algorithm to find the best food source. These were the employed bee, onlooker bee, and scout bee phases. In SOS, new solution generation is governed by imitating the biological interaction between two organisms in the ecosystem. Three phases that resemble the real-world biological interaction model are introduced: mutualism phase, commensalism phase, and parasitism phase.

The character of the interaction defines the main principle of each phase. Interactions benefit both sides in the mutualism phase; benefit one side and do not impact the other in the commensalism phase; benefit one side and actively harm the other in the parasitism phase. Each organism interacts with the other organism randomly through all phases. The process is repeated until termination criteria are met. The following algorithm outline reflects the above explanation:

- Initialization
- REPEAT
  - Mutualism phase
  - Commensalism phase
  - Parasitism phase
- UNTIL (termination criterion is met)

Fig. 2 describes detailed SOS algorithm procedures and the next section provides further details on the three phases.

2.2.1. Mutualism phase

An example of mutualism, which benefits both organism participants, is the relationship between bees and flowers. Bees fly amongst flowers, gathering nectar to turn into honey – an activity that benefits bees. This activity also benefits flowers because bees distribute pollen in the process, which facilitates pollination. This SOS phase mimics such mutualistic relationships.

In SOS, X_i is an organism matched to the ith member of the ecosystem. Another organism X_j is then selected randomly from the ecosystem to interact with X_i. Both organisms engage in a mutualistic relationship with the goal of increasing mutual survival advantage in the ecosystem. New candidate solutions for X_i and X_j are calculated based on the mutualistic symbiosis between organism X_i and X_j, which is modeled in Eqs. (1) and (2).

$$X_{inew} = X_i + rand(0,1) * (X_{best} - Mutual\_Vector * \mathrm{BF}_1) \quad (1)$$

$$X_{jnew} = X_j + rand(0,1) * (X_{best} - Mutual\_Vector * \mathrm{BF}_2) \quad (2)$$

$$Mutual\_Vector = \frac{X_i + X_j}{2} \quad (3)$$

$rand(0,1)$ in Eqs (1) and (2) is a vector of random numbers.

The role of BF_1 and BF_2 is explained as follows. In nature, some mutualism relationships might give a greater beneficial advantage for just one organism than another organism. In other words,

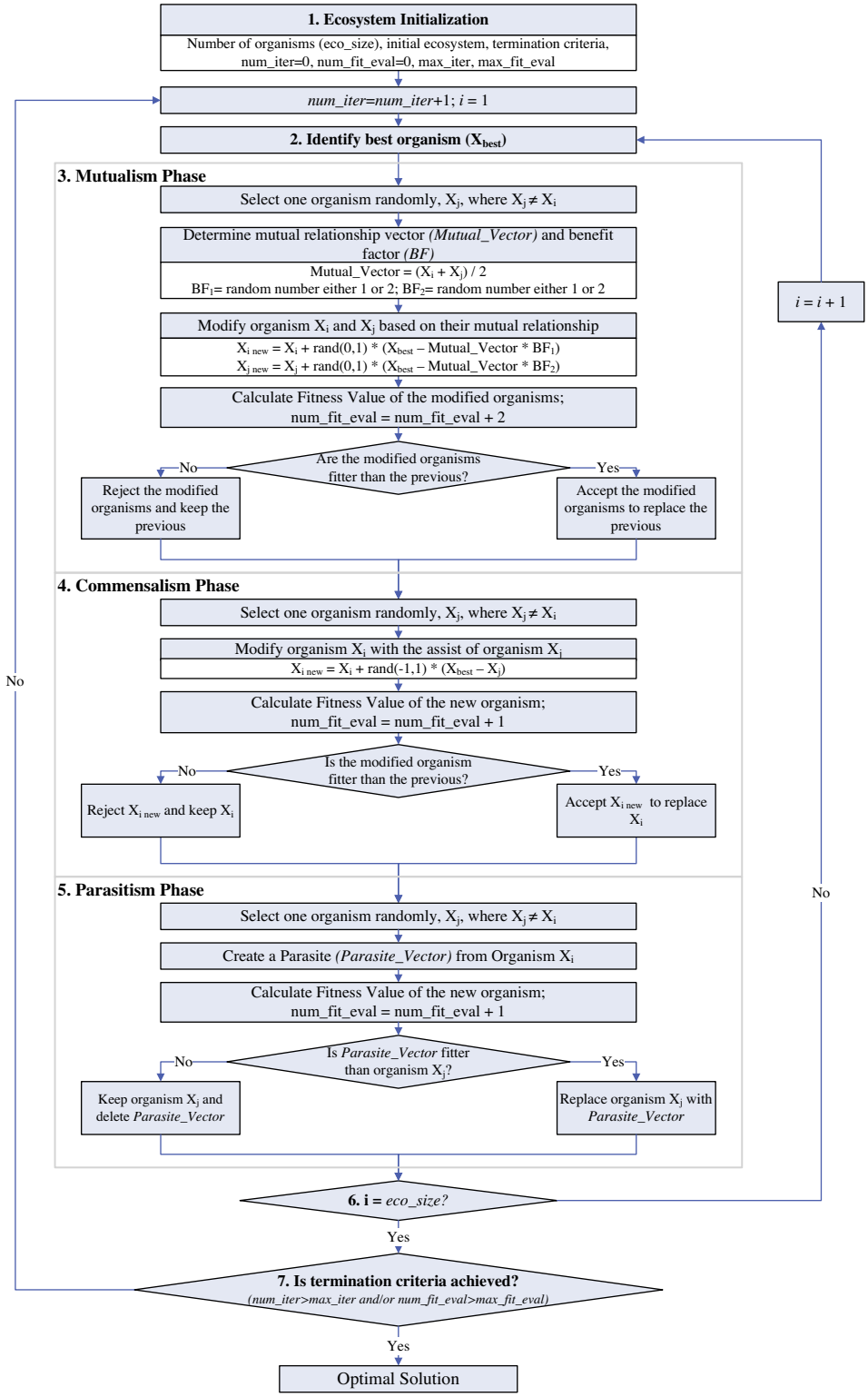

Fig. 2. SOS flowchart.

organism A might receive a huge benefit when interacting with organism B. Meanwhile, organism B might only get adequate or not so significant benefit when interacting with organism A. Here, benefit factors (BF_1 and BF_2) are determined randomly as either 1 or 2. These factors represent level of benefit to each organisms, i.e., whether an organism partially or fully benefits from the interaction.

Eq. (3) shows a vector called "Mutual_Vector" that represents the relationship characteristic between organism X_i and X_j. The part of equation, $(X_{best} - Mutual\_Vector^{*}BF_1)$, is reflecting the mutualistic effort to achieve their goal in increasing their survival advantage. According to the Darwin's evolution theory, "only the fittest organisms will prevail", all creatures are forced to increase their degree of adaptation to their ecosystem. Some of them use

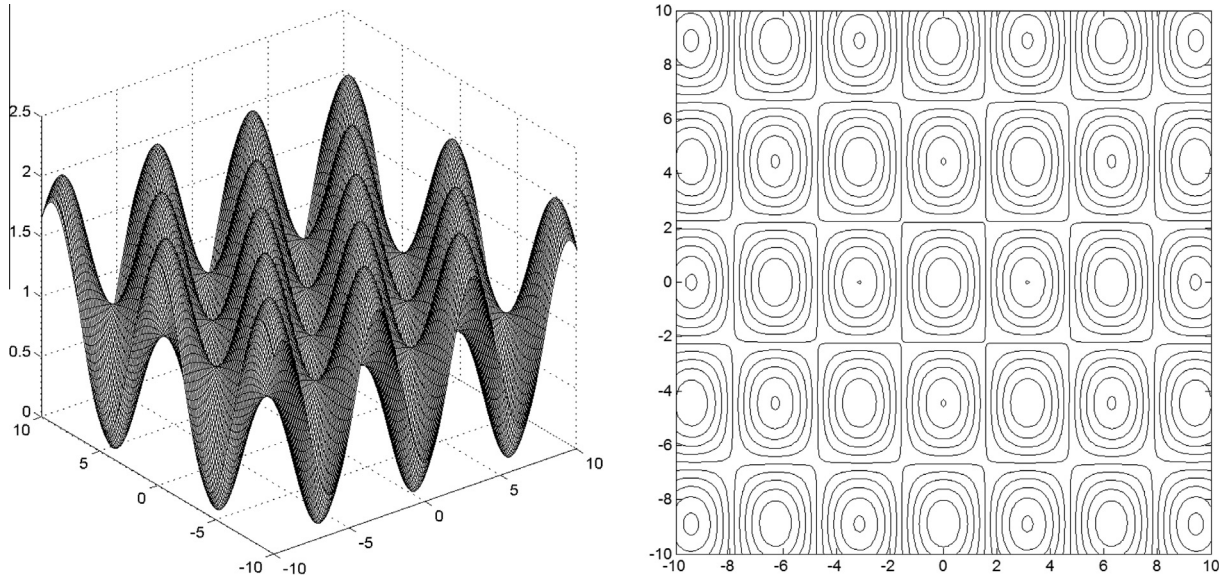

Fig. 3. Three dimensional and contour plots for the Griewank function.

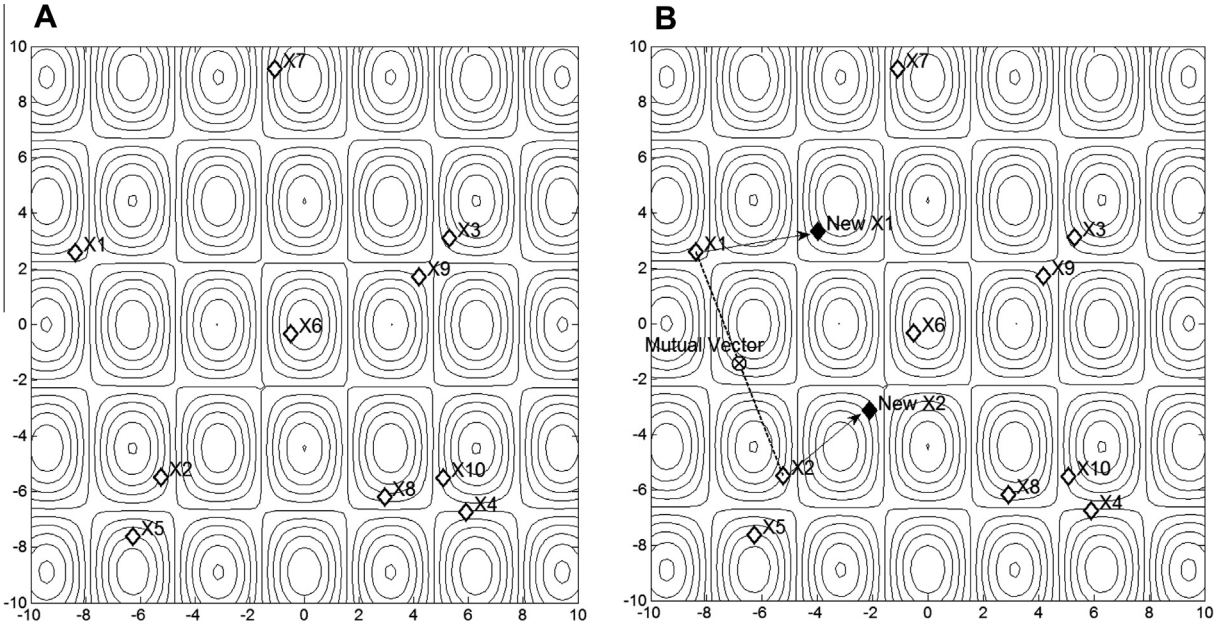

Fig. 4. Visualization of mutualism phase, (A) position in the beginning of phase, (B) position in the end of phase.

symbiotic relationship with others to increase their survival adaptation. The X_{best} is needed here because X_{best} is representing the highest degree of adaptation. Therefore, we use X_{best}/global solution to model the highest degree of adaptation as the target point for the fitness increment of both organisms.

Finally, organisms are updated only if their new fitness is better than their pre-interaction fitness.

2.2.2. Commensalism phase

An example of commensalism is the relationship between remora fish and sharks. The remora attaches itself to the shark and eats food leftovers, thus receiving a benefit. The shark is unaffected by remora fish activities and receives minimal, if any, benefit from the relationship.

Similar to the mutualism phase, an organism, X_j, is selected randomly from the ecosystem to interact with X_i. In this circumstance, organism X_i attempts to benefit from the interaction. However, organism X_j itself neither benefits nor suffers from the relationship. The new candidate solution of X_i is calculated according to the commensal symbiosis between organism X_i and X_j, which is modeled in Eq. (4). Following the rules, organism X_i is updated only if its new fitness is better than its pre-interaction fitness.

$$X_{inew} = X_i + rand(-1,1)^*(X_{best} - X_j) \tag{4}$$

The part of equation, $(X_{best} - X_j)$, is reflecting as the beneficial advantage provided by X_j to help X_i increasing its survival advantage in ecosystem to the highest degree in current organism (represented by X_{best}).

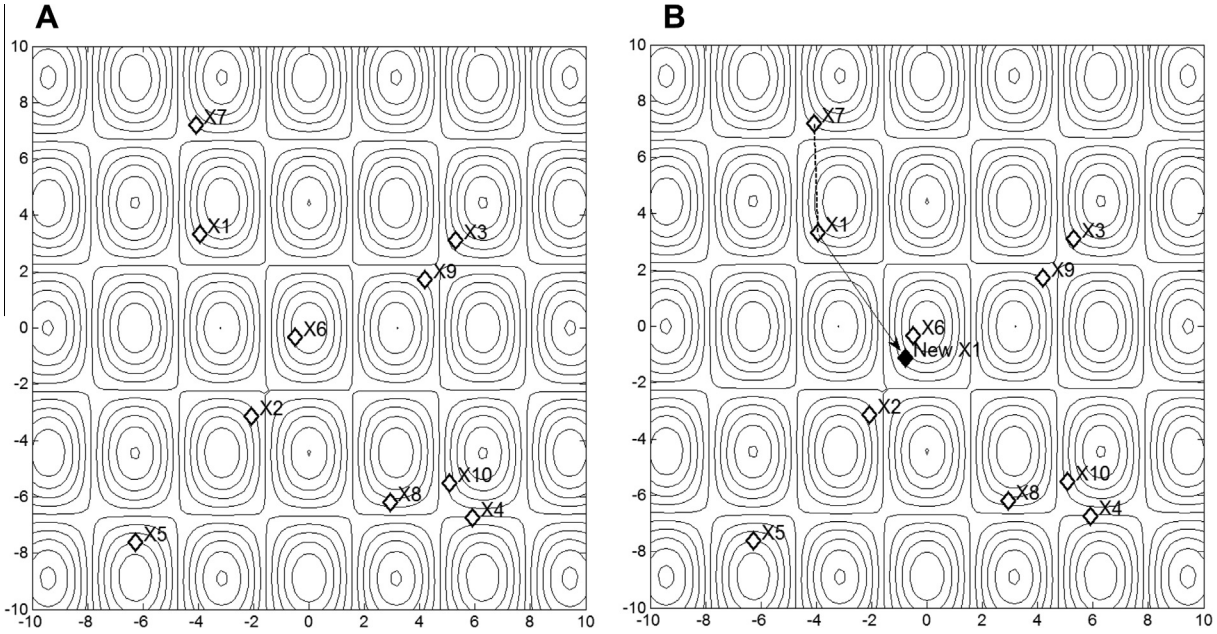

Fig. 5. Visualization of commensalism phase, (A) position in the beginning of phase, (B) position in the end of phase.

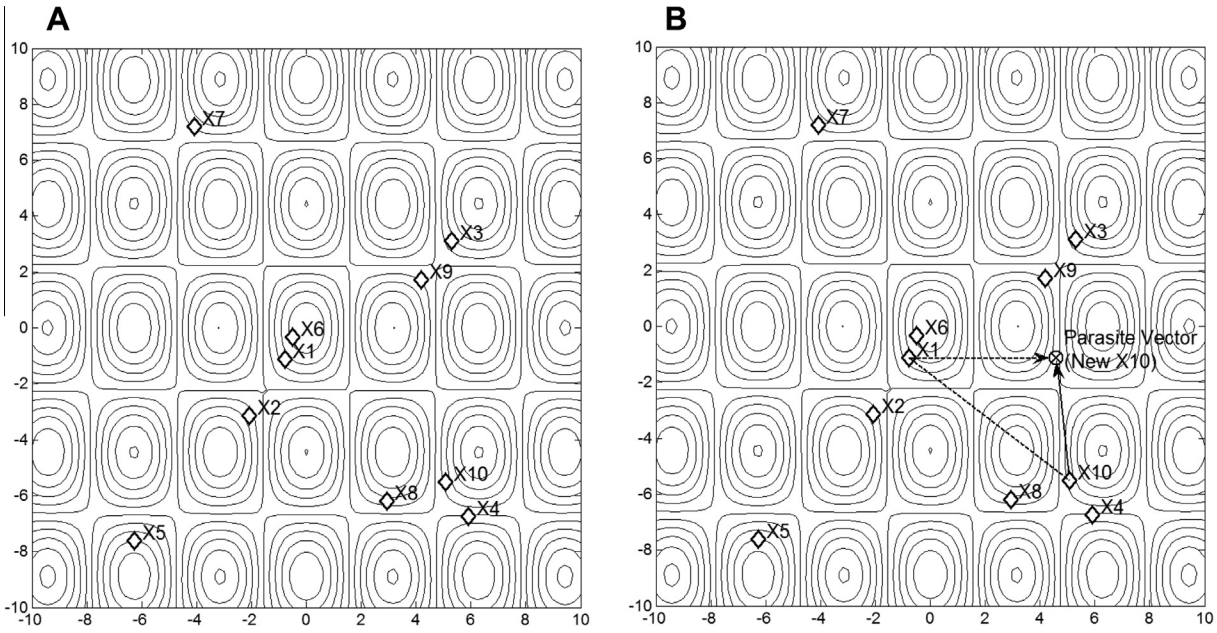

Fig. 6. Visualization of parasitism phase, (A) position in the beginning of phase, (B) position in the end of phase.

2.2.3. Parasitism phase

An example of parasitism is the plasmodium parasite, which uses its relationship with the anopheles mosquito to pass between human hosts. While the parasite thrives and reproduces inside the human body, its human host suffers malaria and may die as a result.

In SOS, organism X_i is given a role similar to the anopheles mosquito through the creation of an artificial parasite called "Parasite_Vector". Parasite_Vector is created in the search space by duplicating organism X_i, then modifying the randomly selected dimensions using a random number. Organism X_j is selected randomly from the ecosystem and serves as a host to the parasite vector. Parasite_Vector tries to replace X_j in the ecosystem. Both organisms are then evaluated to measure their fitness. If Parasite_Vector has a better fitness value, it will kill organism X_j and assume its position in the ecosystem. If the fitness value of X_j is better, X_j will have immunity from the parasite and the Parasite_Vector will no longer be able to live in that ecosystem.

2.3. Implementation of SOS algorithm for numerical optimization

This section introduces the step-wise procedure for implementing SOS. One mathematical function, the Griewank function, was chosen to demonstrate the step-by-step procedure used by SOS to solve numerical problems.

$$f(x) = \frac{1}{4000}\left(\sum_{i=1}^{D}(x_i - 100)^2\right) - \left(\prod_{i=1}^{D}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right)\right) + 1 \qquad (5)$$

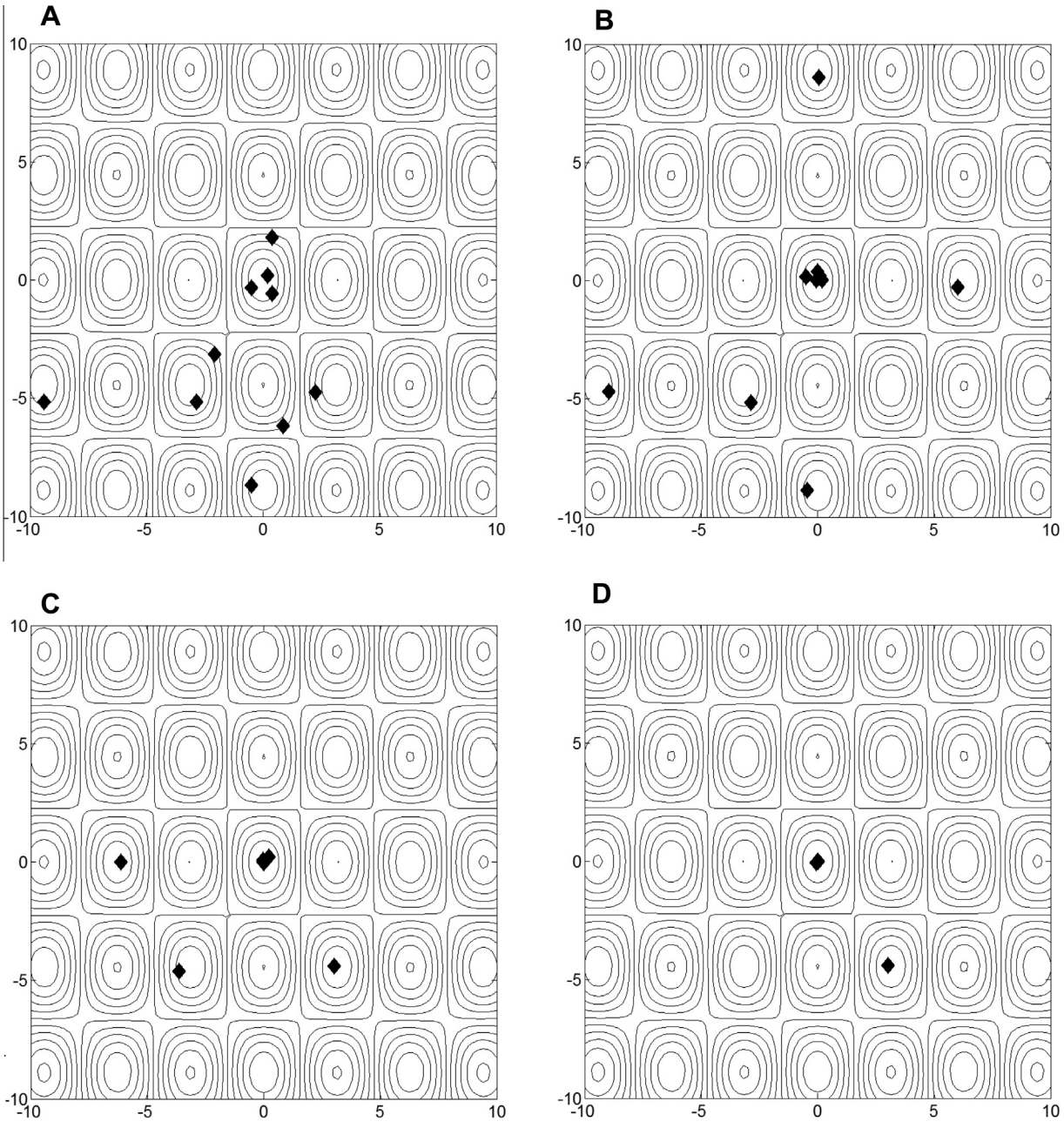

Fig. 7. Visualization of convergence process for Griewank function, (A) Iteration 1, (B) Iteration 5, (C) Iteration 10, (D) Iteration 15.

Griewank function is a multimodal, separable, and regular function with a global minimum $f_{min} = 0$ at $(0,0,\ldots,0)$. Initialization range and number of design variables (D) for this function were set at $[-10,10]$ and 2, respectively. Fig. 3 shows Griewank function three-dimensional and contour plots.

Step 1: Ecosystem initialization

Optimization parameters considered for Griewank functions include:

- Number of organisms (eco_size) = 10.
- Maximum iteration (max_iter) = 20.

Reaching the maximum iteration and finding a global minimum below 1E−06 are the stopping criteria. The initial ecosystem is defined by generating a uniform random number between lower and upper ecosystem size (eco_size) values and a design variable (D) number. This ecosystem is expressed as follows:

$$\text{Ecosystem} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \\ X_9 \\ X_{10} \end{bmatrix} = \begin{bmatrix} -8.3688 & 2.5941 \\ -5.2251 & -5.4795 \\ 5.3103 & 3.1020 \\ 5.9040 & -6.7478 \\ -6.2625 & -7.6200 \\ -0.5047 & -0.3327 \\ -1.0883 & 9.1949 \\ 2.9263 & -6.1923 \\ 4.1873 & 1.7054 \\ 5.0937 & -5.5238 \end{bmatrix}$$

Table 1
The detailed of benchmark functions (D: dimensions, M: multimodal, N: non-separable, U: unimodal, S: separable).

Number	Function	Range	D	Type	Formulation	Min
1	Beale	$[-4.5, 4.5]$	2	UN	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	0
2	Easom	$[-100, 100]$	2	UN	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2 - (x_2-\pi)^2)$	−1
3	Matyas	$[-10, 10]$	2	UN	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	0
4	Bohachevsky1	$[-100, 100]$	2	MS	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	0
5	Booth	$[-10, 10]$	2	MS	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	0
6	Michalewicz2	$[0, \pi]$	2	MS	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(ix_i^2/\pi))^{20}$	−1.8013
7	Schaffer	$[-100, 100]$	2	MN	$f(x) = 0.5 + \frac{\sin^2(\sqrt{x_1^2+x_2^2}) - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$	0
8	Six Hump Camel Back	$[-5, 5]$	2	MN	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	−1.03163
9	Boachevsky2	$[-100, 100]$	2	MN	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$	0
10	Boachevsky3	$[-100, 100]$	2	MN	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$	0
11	Shubert	$[-10, 10]$	2	MN	$f(x) = (\sum_{i=1}^{5} i\cos(i+1)x_1 + i)(\sum_{i=1}^{5} i\cos((i+1)x_2 + i))$	−186.73
12	Colville	$[-10, 10]$	4	UN	$f(x) = 100(x_1^2 - x_2)^2 + (x_1-1)^2 + (x_3-1)^2 + 90(x_3^2 - x_4)^2 + 10.1(x_2-1)^2 + (x_4-1)^2 + 19.8(x_2-1)(x_4-1)$	0
13	Michalewicz5	$[0, \pi]$	5	MS	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(ix_i^2/\pi))^{20}$	−4.6877
14	Zakharov	$[-5, 10]$	10	UN	$f(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5ix_i)^2 + (\sum_{i=1}^{D} 0.5ix_i)^4$	0
15	Michalewicz10	$[0, \pi]$	10	MS	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(ix_i^2/\pi))^{20}$	−9.6602
16	Step	$[-5.12, 5.12]$	30	US	$f(x) = \sum_{i=1}^{D} (x_i + 0.5)^2$	0
17	Sphere	$[-100, 100]$	30	US	$f(x) = \sum_{i=1}^{D} x_i^2$	0
18	SumSquares	$[-10, 10]$	30	US	$f(x) = \sum_{i=1}^{D} ix_i^2$	0
19	Quartic	$[-1.28, 1.28]$	30	US	$f(x) = \sum_{i=1}^{D} ix_i^4 + Rand$	0
20	Schwefel 2.22	$[-10, 10]$	30	UN	$f(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	0
21	Schwefel 1.2	$[-100, 100]$	30	UN	$f(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	0
22	Rosenbrock	$[-30, 30]$	30	MN	$f_5(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	0
23	Dixon-Price	$[-10, 10]$	30	UN	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_i - 1)^2$	0
24	Rastrigin	$[-5.12, 5.12]$	30	MS	$f(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	0
25	Griewank	$[-600, 600]$	30	MN	$f(x) = \frac{1}{4000}\left(\sum_{i=1}^{D} (x_i - 100)^2\right) - \left(\prod_{i=1}^{D} \cos\left(\frac{x_i - 100}{\sqrt{i}}\right)\right) + 1$	0
26	Ackley	$[-32, 32]$	30	MN	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$	0

$$\text{and the corresponding fitness value} = \begin{bmatrix} 0.8910 \\ 1.3789 \\ 1.3377 \\ 0.9653 \\ 0.3989 \\ 0.1489 \\ 0.5685 \\ 0.6917 \\ 1.1840 \\ 1.2828 \end{bmatrix}$$

Step 2: Identify the best solution, X_{best}.
X_6 has the lowest minimum fitness value of the entire ecosystem and is chosen as X_{best}.

Step 3: Mutualism phase
with i set initially at 1, organism X_1 is matched to X_i. Meanwhile, organism X_j is selected randomly from the ecosystem. In this case, X_2 is selected as X_j.

Mutual_Vector is determined using Eq. (3). Benefit Factors (BF_1 and BF_2) are determined by randomly assigning values of either 1 or 2.

- Mutual_Vector = [−6.7970, −1.4427].
- BF_1 = 1 and BF_2 = 2.
  New candidate solutions X_1 and X_2 are calculated using Eqs. (1) and (2).
- $X_{1new} = X_1 + \text{rand}(0,1) * (X_6 - \text{Mutual\_Vector} * BF_1)$.
- $X_{2new} = X_2 + \text{rand}(0,1) * (X_6 - \text{Mutual\_Vector} * BF_2)$.
- X_{1new} = [−3.9498, 3.3352] with the corresponding fitness value = [0.5172].
- X_{2new} = [−2.0879, −3.1486] with the corresponding fitness value = [0.7022].
  New candidate solutions X_1 and X_2 are compared to the old X_1 and X_2. Fitter organisms are selected as solutions for the next iteration.
- New X_1 fitness is better than the old. Therefore, X_1 is modified to [−3.9498, 3.3352].

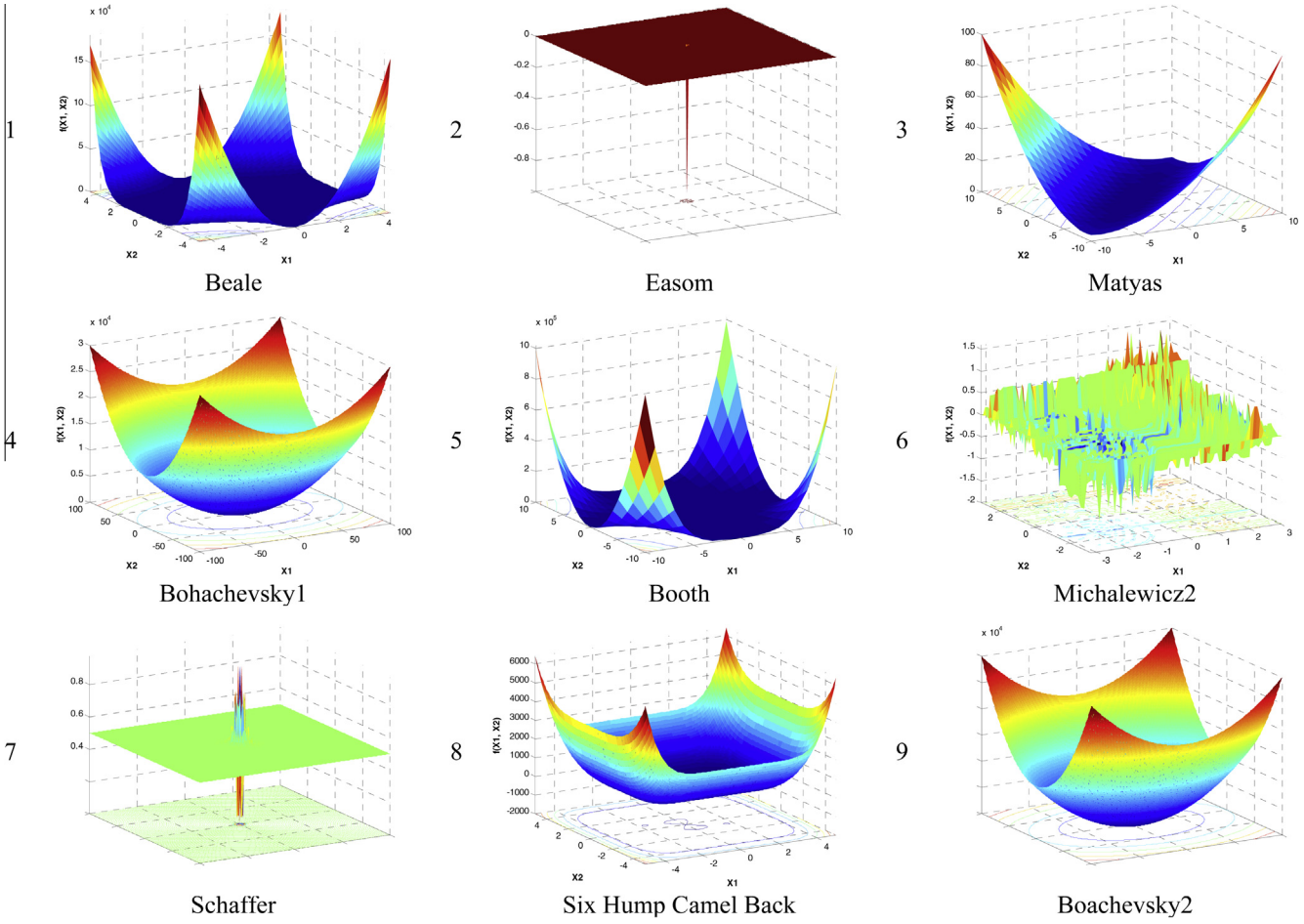

Fig. 8. A perspective view for functions 1 to 9.

- New X_2 fitness is better than the old Therefore, X_2 is modified to [−2.0879, −3.1486].
  Fig. 4 visualizes the interaction pattern of mutualism in the search space.

Step 4: Commensalism phase

Organism X_7 is selected randomly from the ecosystem. New candidate solutions X_1 are calculated using Eq. (4).

- $X_{1new} = X_1 + \text{rand}(-1,1)^*(X_6 - X_7)$.
- X_{1new} = [−0.7887, −1.1078] with the corresponding fitness value = [0.5011].
  New candidate solution X_1 is compared to the older X_1. The fitter organism is chosen as the solution for the next iteration.
- New X_1 fitness > old X_1 fitness. Therefore, X_1 is modified to [−0.7887, −1.1078].
  Fig. 5 visualizes the parasitism mechanism in the search space.

Step 5: Parasitism phase

Organism X_{10} is selected randomly from the ecosystem. Parasite_Vector is created by mutating X_i in random dimensions using a random number with a range between given lower and upper bounds.

- Parasite_Vector = [4.5782, −1.1078] with the corresponding fitness value = [1.103].
  The Parasite_Vector is compared to X_{10}. The fitter organism will survive to the next iteration.
- The fitness of Parasite_Vector is better than X_{10}. Therefore, X_{10} is eliminated from the ecosystem and replaced by parasite vector = [4.5782, −1.1078].
  Fig. 6 visualizes the interaction pattern of mutualism in the search space.

Step 6: Go to step 2 if the current X_i is not the last member of the ecosystem; otherwise proceed to next step.

Step 7: Stop if one of the termination criteria is reached; otherwise return to step 2 and start the next iteration.

Although the Griewank function is bowl-shaped, finding its global optimum is still challenging due to many local minima in the search space. According to Whitley et al., this function is more difficult to solve in lower dimensions than higher dimensions [23]. Surprisingly, SOS found the global optimum before the maximum iteration was achieved. The best solution with a fitness value of 4.87E−8 was found after 15 iterations. Fig. 7 visualizes the ecosystem converging toward the global optima. Further, more than half of the organisms had already approached the global optimum valley after just 5 iterations and begun converging on the optimum point, as shown in Fig. 7.

3. SOS validation

This study considered several numerical optimization problems from the literature to validate SOS performance. This section is divided into two sub-sections. Section 3.1 provides a large set of complex mathematical benchmark problems to be tested, with results compared against other metaheuristic algorithms. Section 3.2 examines six structural engineering design problems.

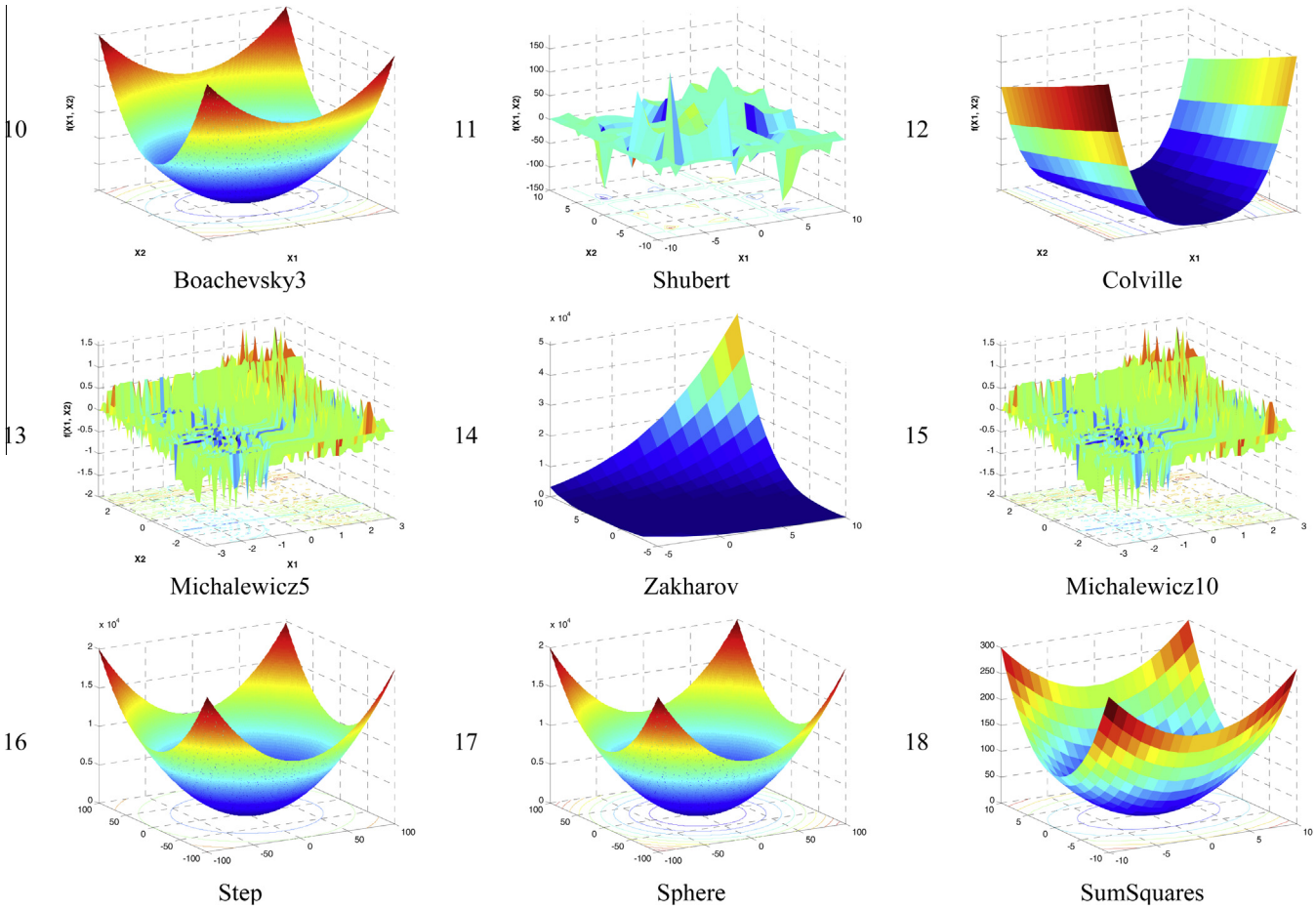

Fig. 9. A perspective view for functions 10 to 18.

3.1. Mathematical benchmark problems

This section compares the performance of SOS to the performance of other metaheuristic algorithms including GA, DE, PSO, BA, and the hybrid PSO-BA (Particle Bee Algorithm [PBA]) [24] using 26 metaheuristic algorithm benchmark functions described by Cheng and Lien [24]. Functions 1–11 are two-dimensional; functions 12 and 13 are four- and five-dimensional; and functions 16–26 are 30-dimensional. All functions may be separated into the type categories of multimodal/unimodal and separable/non-separable. Table 1 provides benchmark function details. Perspective views on these benchmarks are shown in Figs. 8–10.

Cheng and Lien [24] previously conducted experiments on all functions with a 500,000 maximum number of function evaluations. They reported any value less than 1E−12 as 0. To maintain comparison consistency, SOS was also tested using these same conditions. Table 2 lists control and specific parameter settings for each algorithm.

Table 3 delineates the respective performance of SOS and other algorithms in solving benchmark functions. Performance values for all algorithms except for SOS reference Cheng and Lien [24]. The mean value and standard deviation for SOS were obtained after 30 independent runs, in line with standards followed in the previous work. In Table 3, bolded numbers represent the comparatively best values. SOS found the global optimum value for 22 of the 26 functions and outperformed all other algorithms tested. Further, SOS was the only algorithm able to solve Dixon-Price (function 23) and produced the best result of all on the exceptionally difficult Rosenbrock (function 22).

3.2. Engineering design problems

This section examines SOS performance using six engineering design optimization problems from the structural engineering field, with SOS optimization results compared to data published in the literature. SOS used 20 organisms for all cases. Different maximum numbers of function evaluations were used for each problem, with smaller function evaluation numbers used for smaller number of design variables and moderate functions and larger function evaluation numbers used for larger design variable numbers and complex problems. As for constraint handling method, Deb's feasibility rules is considered in this study [25]. Because SOS is parameter-free, only common control parameters (population size and maximum number of function evaluations) were adjusted. Therefore, SOS performance was consistent across different problems.

3.2.1. Cantilever beam

The cantilever beam problem was adopted from Chickermane and Gea [26]. The cantilever beam shown in Fig. 3 comprises five elements. Each element has a hollow cross section of a fixed diameter. The beam is rigidly supported as shown, and a vertical force acts at the free end of the cantilever. The problem presented is to minimize beam weight. The design variable is the height (or width) x_i of each beam element. Bound constraints are set as $0.01 \leqslant x_i \leqslant 100$. The problem is formulated using classical beam theory as (see Fig. 11):

$$\text{Minimize } f(X) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5) \tag{6}$$

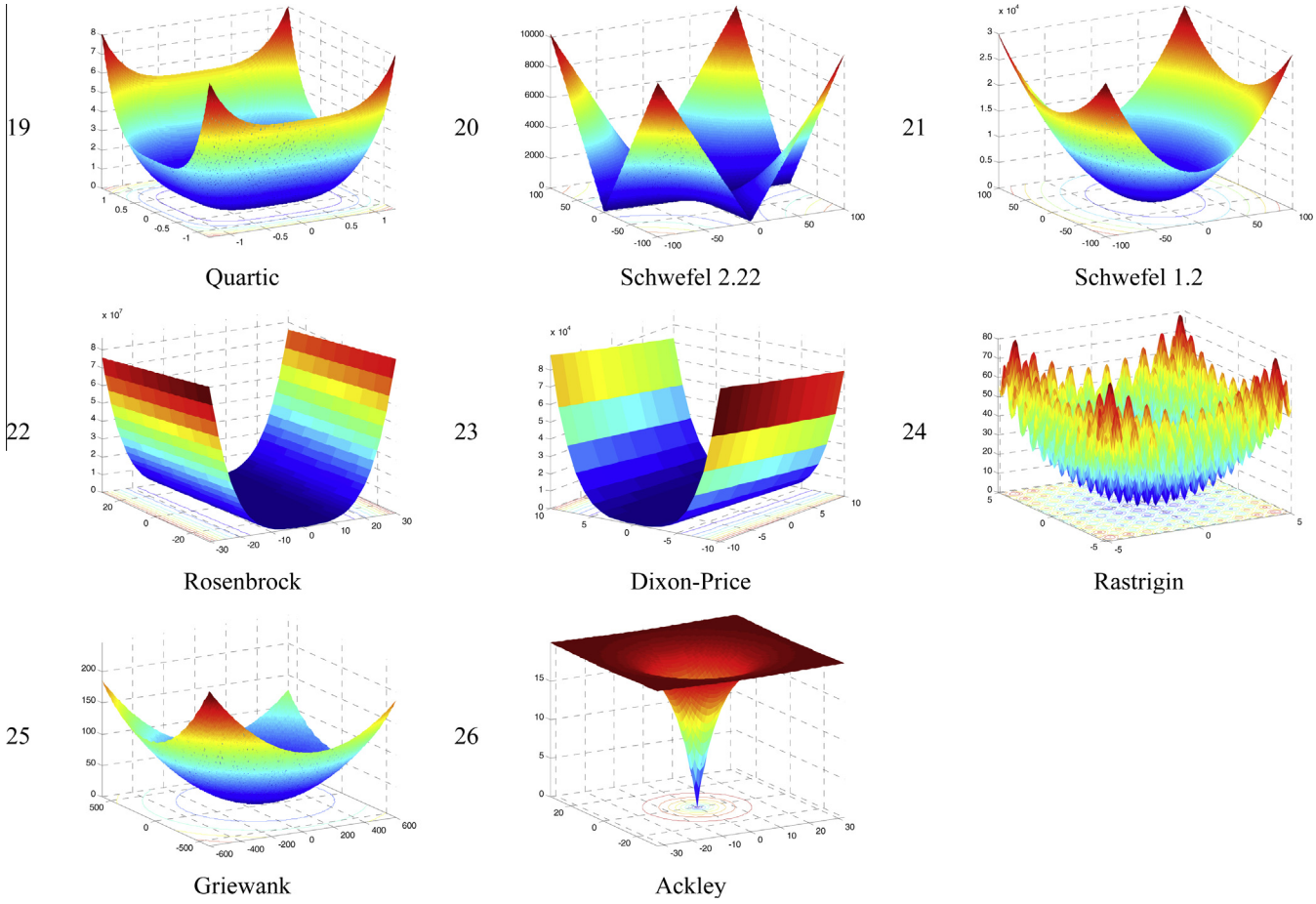

Fig. 10. A perspective view for functions 19 to 26.

Table 2
Parameter settings of the algorithms.

GA	DE	PSO	BA	PBA	SOS
$n = 50$ $m = 0.01$ $c = 0.8$ $g = 0.9$	$n = 50$ $c = 0.9$ $F = 0.5$	$n = 50$ $w = 0.9–0.7$ $v = X_{min}/10 \sim X_{max}/10$	$n = 50$ $e = NP/2$ $b = NP/4$ $r = NP/4$ $n_1 = 2$ $n_2 = 1$	$n = 50$ $e = NP/2$ $b = NP/4$ $r = NP/4$ $w = 0.9–0.7$ $v = X_{min}/10 \sim X_{max}/10$ *Pelite* = 15 *Pbest* = 9	$n = 50$

Note: n = population size/colony size/ecosystem size; m = mutation rate; c = crossover rate; g = generation gap; f = scaling factor; w = inertia weight; v = limit of velocity; e = elite bee number; b = best bee number; r = random bee number; n_1 = elite bee neighborhood number; n_2 = best bee neighborhood number; *Pelite* = PSO iteration of elite bees; *Pbest* = PSO iteration of best bees.

Subject to:

$$g(X) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \leqslant 1 \tag{7}$$

Table 4 lists the best solutions obtained by SOS and various methods [26,27]. SOS achieved a solution superior to all other methods.

3.2.2. Minimize I-beam vertical deflection

This case study, using a design problem with four variables, was modified from an original problem reported in [28]. Fig. 12 illustrates the goal of this case – minimizing the vertical deflection of an I-beam. The cross-sectional area and stress constraints must be simultaneously satisfied under given loads.

The objective description of this study is to minimize the vertical deflection $f(x) = PL^3/48EI$ when beam length (L) and elasticity modulus (E) are, respectively, 5200 cm and 523,104 kN/cm^2. Thus, the objective function of the problem is considered to be:

$$\text{Minimize}: \ f(b,h,t_w,t_f) = \frac{5000}{\frac{t_w(h-2t_f)}{12} + \frac{bt_f^3}{6} + 2bt_f\left(\frac{h-t_f}{2}\right)^2} \tag{8}$$

Subject to a cross-section area less than 300 cm^2

$$g_1 = 2bt_w + t_w(h-2t_f) \leqslant 300 \tag{9}$$

If allowable bending stress of the beam is 56 kN/cm^2, the stress constraint is:

Table 3
Comparative results of SOS with GA, DE, PSO, BA, and PBA (Bolded numbers represent the best values).

No.	Functions		Min	GA	DE	PSO	BA	PBA	SOS
1	Beale	Mean	0	**0**	**0**	**0**	1.88E−05	**0**	**0**
		StdDev		**0**	**0**	**0**	1.94E−05	**0**	**0**
2	Easom	Mean	−1	**−1**	**−1**	**−1**	−0.99994	**−1**	**−1**
		StdDev		**0**	**0**	**0**	4.50E−05	**0**	**0**
3	Matyas	Mean	0	**0**	**0**	**0**	**0**	**0**	**0**
		StdDev		**0**	**0**	**0**	**0**	**0**	**0**
4	Bohachevsky1	Mean	0	**0**	**0**	**0**	**0**	**0**	**0**
		StdDev		**0**	**0**	**0**	**0**	**0**	**0**
5	Booth	Mean	0	**0**	**0**	**0**	0.00053	**0**	0.03382
		StdDev		**0**	**0**	**0**	0.00074	**0**	0.12870
6	Michalewicz2	Mean	−1.8013	**−1.8013**	**−1.8013**	−1.57287	**−1.8013**	**−1.8013**	**−1.8013**
		StdDev		**0**	**0**	0.11986	**0**	**0**	**0**
7	Schaffer	Mean	0	0.00424	**0**	**0**	**0**	**0**	**0**
		StdDev		0.00476	**0**	**0**	**0**	**0**	**0**
8	Six Hump Camel Back	Mean	−1.03163	**−1.03163**	**−1.03163**	**−1.03163**	**−1.03163**	**−1.03163**	**−1.03163**
		StdDev		**0**	**0**	**0**	**0**	**0**	**0**
9	Boachevsky2	Mean	0	0.06829	**0**	**0**	**0**	**0**	**0**
		StdDev		0.07822	**0**	**0**	**0**	**0**	**0**
10	Boachevsky3	Mean	0	**0**	**0**	**0**	**0**	**0**	**0**
		StdDev		**0**	**0**	**0**	**0**	**0**	**0**
11	Shubert	Mean	−186.73	**−186.73**	**−186.73**	**−186.73**	**−186.73**	**−186.73**	**−186.73**
		StdDev		**0**	**0**	**0**	**0**	**0**	**0**
12	Colville	Mean	0	0.01494	0.04091	**0**	1.11760	**0**	**0**
		StdDev		0.00736	0.08198	**0**	0.46623	**0**	**0**
13	Michalewicz5	Mean	−4.6877	−4.64483	−4.68348	−2.49087	**−4.6877**	**−4.6877**	**−4.6877**
		StdDev		0.09785	0.01253	0.25695	**0**	**0**	**0**
14	Zakharov	Mean	0	0.01336	**0**	**0**	**0**	**0**	**0**
		StdDev		0.00453	**0**	**0**	**0**	**0**	**0**
15	Michalewicz10	Mean	−9.6602	−9.49683	−9.59115	−4.00718	**−9.6602**	**−9.6602**	−9.65982
		StdDev		0.14112	0.06421	0.50263	**0**	**0**	0.00125
16	Step	Mean	0	1.17E+03	**0**	**0**	5.1237**0**	**0**	**0**
		StdDev		76.56145	**0**	**0**	0.39209	**0**	**0**
17	Sphere	Mean	0	1.11E+03	**0**	**0**	**0**	**0**	**0**
		StdDev		74.21447	**0**	**0**	**0**	**0**	**0**
18	SumSquares	Mean	0	1.48E+02	**0**	**0**	**0**	**0**	**0**
		StdDev		12.40929	**0**	**0**	**0**	**0**	**0**
19	Quartic	Mean	0	0.18070	0.00136	0.00116	1.72E−06	0.00678	9.13E−05
		StdDev		0.02712	0.00042	0.00028	1.85E−06	0.00133	3.71E−05
20	Schwefel 2.22	Mean	0	11.0214	**0**	**0**	**0**	7.59E−10	**0**
		StdDev		1.38686	**0**	**0**	**0**	7.10E−10	**0**
21	Schwefel 1.2	Mean	0	7.40E+03	**0**	**0**	**0**	**0**	**0**
		StdDev		1.14E+03	**0**	**0**	**0**	**0**	**0**
22	Rosenbrock	Mean	0	1.96E+05	18.20394	15.088617	28.834	4.2831	1.04E−07
		StdDev		3.85E+04	5.03619	24.170196	0.10597	5.7877	2.95E−07
23	Dixon−Price	Mean	0	1.22E+03	0.66667	0.66667	0.66667	0.66667	**0**
		StdDev		2.66E+02	E−9	E−8	1.16E−09	5.65E−10	**0**
24	Rastrigin	Mean	0	52.92259	11.71673	43.97714	**0**	**0**	**0**
		StdDev		4.56486	2.53817	11.72868	**0**	**0**	**0**
25	Griewank	Mean	0	10.63346	0.00148	0.01739	**0**	0.00468	**0**
		StdDev		1.16146	0.00296	0.02081	**0**	0.00672	**0**
26	Ackley	Mean	0	14.67178	**0**	0.16462	**0**	3.12E−08	**0**
		StdDev		0.17814	**0**	0.49387	**0**	3.98E−08	**0**
Count of algorithm found global minimum				9	18	17	18	20	22

$$g_2 = \frac{18h \times 10^4}{t_w(h-2t_f)^3 + 2bt_w(4t_f^2 + 3h(h-2t_f))} + \frac{15b \times 10^3}{(h-2t_f)t_w^3 + 2t_w b^3} \leqslant 56 \quad (10)$$

where initial design spaces are $10 \leqslant h \leqslant 80$, $10 \leqslant b \leqslant 50$, $0.9 \leqslant t_w \leqslant 5$, and $0.9 \leqslant t_f \leqslant 5$.

For this case study, 5000 completed function evaluations was set as the stopping criterion. Table 5 presents results obtained by SOS and other algorithms. This case study has been previously solved using other methods such as adaptive surface method (ARSM), improved ARSM [29], and CS [27]. SOS performance surpassed ARSM and improved ARSM in terms of both minimum obtained value and solution average. Although CS matched the results obtained using SOS, SOS was still slightly better in terms of mean and standard deviation.

3.2.3. A 15-bar planar truss structure

Fig. 13 shows the geometry and the loading condition of the planar truss structures consisting of 15 bars. This structure was previously optimized by [30,31]. The material density is 7800 kg/m^3 and the modulus of elasticity is 200 GPa. The maximum allowable stress for all members is 120 MPa (the same for tension and compression) and maximum displacements for all free nodes in the X and Y directions must not exceed 10 mm. There are 15 design variables in this problem. We selected discrete variables from the set:

D = [113.2, 143.2, 145.9, 174.9, 185.9, 235.9, 265.9, 297.1, 308.6, 334.3, 338.2, 497.8, 507.6, 736.7, 791.2, 1063.7] (mm^2). This structure is subjected to three load cases as follows: Load Case 1:

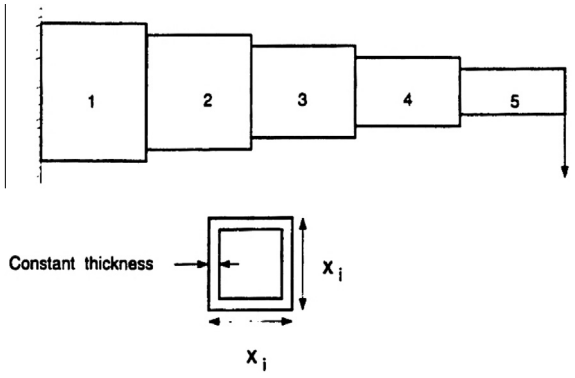

Fig. 11. Cantilever beam problem [27].

Table 4
Best solution for the cantilever beam design.

	Chickermane and Gea [26]				Gandomi et al. [27]	Present Study
	CONLIN	MMA	GCA(I)	GCA(II)	CS	SOS
x_1	6.0100	6.0100	6.0100	6.0100	6.0089	6.01878
x_2	5.3000	5.3000	5.3000	5.3000	5.3049	5.30344
x_3	4.4900	4.4900	4.4900	4.4900	4.5023	4.49587
x_4	3.4900	3.4900	3.4900	3.4900	3.5077	3.49896
x_5	2.1500	2.1500	2.1500	2.1500	2.1504	2.15564
f_{min}	N.C.[a]	1.3400	1.3400	1.3400	1.33999	1.33996
Average	N.A.	N.A.	N.A.	N.A.	N.A.	1.33997
Standard deviation	N.A.	N.A.	N.A.	N.A.	N.A.	1.1E−5
No. evaluations	N.A.	N.A.	N.A.	N.A.	N.A.	15,000

[a] Not converge.

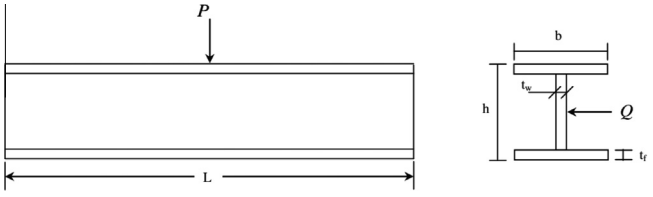

Fig. 12. I-beam design problem [27].

P_1 = 35 kN, P_2 = 35 kN, P_3 = 35 kN; Load Case 2: P_1 = 35 kN, P_2 = 0 kN, P_3 = 35 kN; and Load Case 3: P_1 = 35 kN, P_2 = 35 kN, P_3 = 0 kN.

Table 6 shows the optimal result of the SOS algorithm with 15 size variables and compares these results with those previously reported in the literature. SOS can outperform the Improved-GA, PSO, and PSO while able to equal the performance of HPSO. Meanwhile, HPSO obtained a minimum weight of 105.735 lbs in more than 7500 structural analyses [31], while the SOS algorithm obtained the same in only 5000 structural analyses.

3.2.4. 52-bar planar truss structure

Fig. 14 shows the geometry and loading condition of planar truss structures consisting of fifty-two bars and twenty nodes. This problem was previously been optimized by [32,33,31,34], and [15]. The members of this structure are divided into 12 groups: (1) A_1–A_4, (2) A_5–A_{10}, (3) A_{11}–A_{13}, (4) A_{14}–A_{17}, (5) A_{18}–A_{23}, (6) A_{24}–A_{26}, (7) A_{27}–A_{30}, (8) A_{31}–A_{36}, (9) A_{37}–A_{39}, (10) A_{40}–A_{43}, (11) A_{44}–A_{49}, and (12) A_{50}–A_{52}. The material density is 7860.0 kg/m^3 and the modulus of elasticity is 2.07×10^5 MPa. The maximum allowable stress for all members in terms of both tension and compression is 180 MPa. Loads P_x = 100 kN and P_y = 200 kN are both considered. Discrete variables are selected from Table 7.

Previously, all algorithms were tested using 50 populations and 3000 maximum iterations [15,31,34]. To maintain comparison consistency, SOS uses the same criteria with previous studies. Table 8 shows the results obtained by the SOS algorithm and other optimization methods. It can be observed that HPSO, PSO, PSOPC, and HPSO cannot find a good result, while DHPSACO and SOS algorithms achieve the optimal weight of 1902.605 kg. Figs. 15 and 16 provide convergence capability of the 52-bar planar truss structure. From Figs. 15 and 16, it can be seen that SOS algorithm

Table 5
Best solution for I-beam design.

	Wang [29]		Gandomi et al. [27]	Present Study
	ARSM	Improved ARSM	CS	SOS
h (cm)	80.00	79.99	80.000000	80.00000
b (cm)	37.05	48.42	50.000000	50.00000
t_w (cm)	1.71	0.90	0.900000	0.90000
t_f (cm)	2.31	2.40	2.3216715	2.32179
f_{min} (cm)	0.0157	0.0131	0.0130747	0.0130741
Average	N.A.	N.A.	0.01353646	0.0130884
Standard deviation	N.A.	N.A.	1.3E−4	4.0E−5
No. evaluation	N.A.	N.A.	5,000	5,000

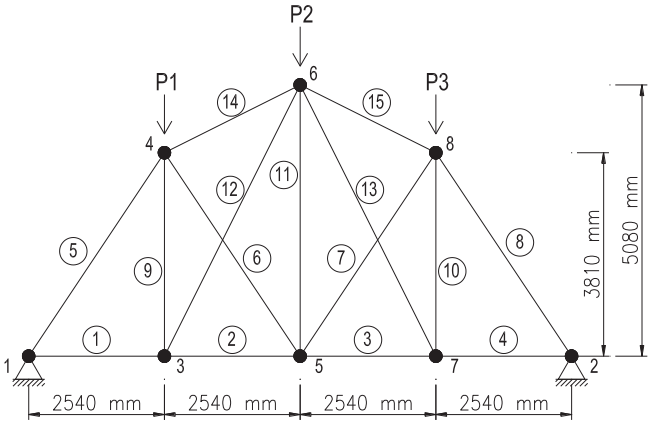

Fig. 13. A 15-bar planar truss structure.

Table 6
Optimal design comparison for the 15-bar planar truss structure.

Variables (mm^2)	Improved-GA [30]	PSO [31]	PSOPC [31]	HPSO [31]	SOS
A_1	308.6	185.9	113.2	113.2	113.2
A_2	174.9	113.2	113.2	113.2	113.2
A_3	338.2	143.2	113.2	113.2	113.2
A_4	143.2	113.2	113.2	113.2	113.2
A_5	736.7	736.7	736.7	736.7	736.7
A_6	185.9	143.2	113.2	113.2	113.2
A_7	265.9	113.2	113.2	113.2	113.2
A_8	507.6	736.7	736.7	736.7	736.7
A_9	143.2	113.2	113.2	113.2	113.2
A_{10}	507.6	113.2	113.2	113.2	113.2
A_{11}	279.1	113.2	113.2	113.2	113.2
A_{12}	174.9	113.2	113.2	113.2	113.2
A_{13}	297.1	113.2	185.9	113.2	113.2
A_{14}	235.9	334.3	334.3	334.3	334.3
A_{15}	265.9	334.3	334.3	334.3	334.3
Weight (kg)	142.117	108.84	108.96	105.735	105.735

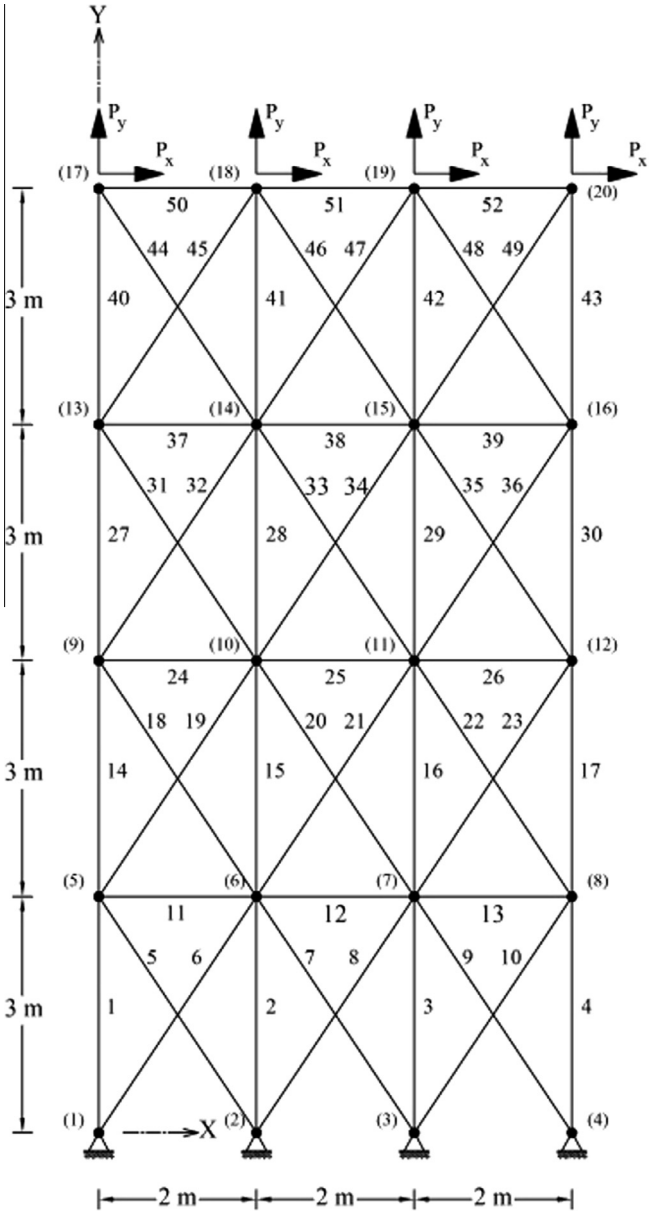

Fig. 14. A 52-bar planar truss structure.

obtained the optimum solution after 47 iterations faster than all. The closest competitor to SOS was MBA, which converged after 212 iterations [15].

Table 7
The available cross-section areas of the AISC code.

No.	in.^{2}	mm^2	No.	in.^{2}	mm^2
1	0.111	71.613	33	3.840	2477.414
2	0.141	90.968	34	3.870	2496.769
3	0.196	126.451	35	3.880	2503.221
4	0.250	161.290	36	4.180	2696.769
5	0.307	198.064	37	4.220	2722.575
6	0.391	252.258	38	4.490	2896.768
7	0.442	285.161	39	4.590	2961.284
8	0.563	363.225	40	4.800	3096.768
9	0.602	388.386	41	4.970	3206.445
10	0.766	494.193	42	5.120	3303.219
11	0.785	506.451	43	5.740	3703.218
12	0.994	641.289	44	7.220	4658.055
13	1.000	645.160	45	7.970	5141.925
14	1.228	792.256	46	8.530	5503.215
15	1.266	816.773	47	9.300	5999.988
16	1.457	939.998	48	10.850	6999.986
17	1.563	1008.385	49	11.500	7419.340
18	1.620	1045.159	50	13.500	8709.660
19	1.800	1161.288	51	13.900	8967.724
20	1.990	1283.868	52	14.200	9161.272
21	2.130	1374.191	53	15.500	9999.980
22	2.380	1535.481	54	16.000	10322.560
23	2.620	1690.319	55	16.900	10903.204
24	2.630	1696.771	56	18.800	12129.008
25	2.880	1858.061	57	19.900	12838.684
26	2.930	1890.319	58	22.000	14193.520
27	3.090	1993.544	59	22.900	14774.164
28	3.130	2019.351	60	24.500	15806.420
29	3.380	2180.641	61	26.500	17096.740
30	3.470	2238.705	62	28.000	18064.480
31	3.550	2290.318	63	30.000	19354.800
32	3.630	2341.931	64	33.500	21612.860

4. Discussion

This study compared SOS algorithm performance against other metaheuristic algorithms such as GA, DE, PSO, BA, and PBA in mathematical benchmark functions and MBA and CS in structural design optimization. SOS performed consistently better than the other algorithms in all benchmark function and optimization problems. This section summarizes the advantages of SOS.

- SOS shares characteristics similar to most population-based algorithms (GA, DE, PSO, MBA, CS, etc.) and performs specific operators iteratively on a group of candidate solutions to achieve a global solution.
- SOS does not reproduce or create children, unlike GA, DE, and numerous other evolutionary algorithms.
- SOS adapts through individual interactions. Although similar in this respect to PSO and DE, the SOS strategy differs significantly.

Table 8
Optimal design comparison for the 52-bar spatial truss structure.

Variables (mm^2)	PSO [31]	PSOPC [31]	HPSO [31]	DHPSACO [31]	MBA [15]	SOS
*A*1–*A*4	4658.055	5999.988	4658.055	4658.055	4658.055	4658.055
*A*5–*A*10	1374.190	1008.380	1161.288	1161.288	1161.288	1161.288
*A*11–*A*13	1858.060	2696.770	363.225	494.193	494.193	494.193
*A*14–*A*17	3206.440	3206.440	3303.219	3303.219	3303.219	3303.219
*A*18–*A*23	1283.870	1161.290	940.000	1008.385	940.000	940.000
*A*24–*A*26	252.260	729.030	494.193	285.161	494.193	494.193
*A*27–*A*30	3303.220	2238.710	2238.705	2290.318	2238.705	2238.705
*A*31–*A*36	1045.160	1008.380	1008.385	1008.385	1008.385	1008.385
*A*37–*A*39	126.450	494.190	388.386	388.386	494.193	494.193
*A*40–*A*43	2341.93	1283.870	1283.868	1283.868	1283.868	1283.868
*A*44–*A*49	1008.38	1161.290	1161.288	1161.288	1161.288	1161.288
*A*50–*A*52	1045.16	494.190	792.256	506.451	494.193	494.193
Weight (kg)	2230.160	2146.630	1905.495	1904.830	1902.605	1902.605

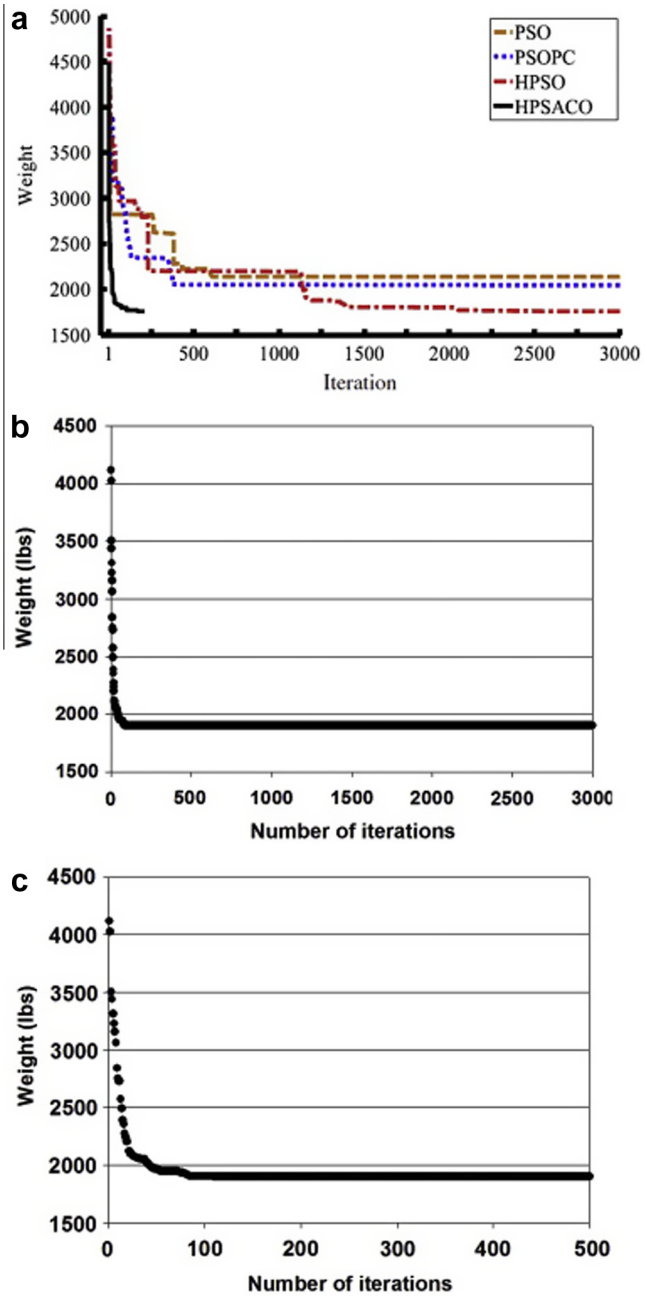

Fig. 15. Comparison of convergence rates for the 52-bar truss using past literatures: (a) DHPSACO, (b) MBA for 3000 iterations and (c) MBA for 500 iterations [15].

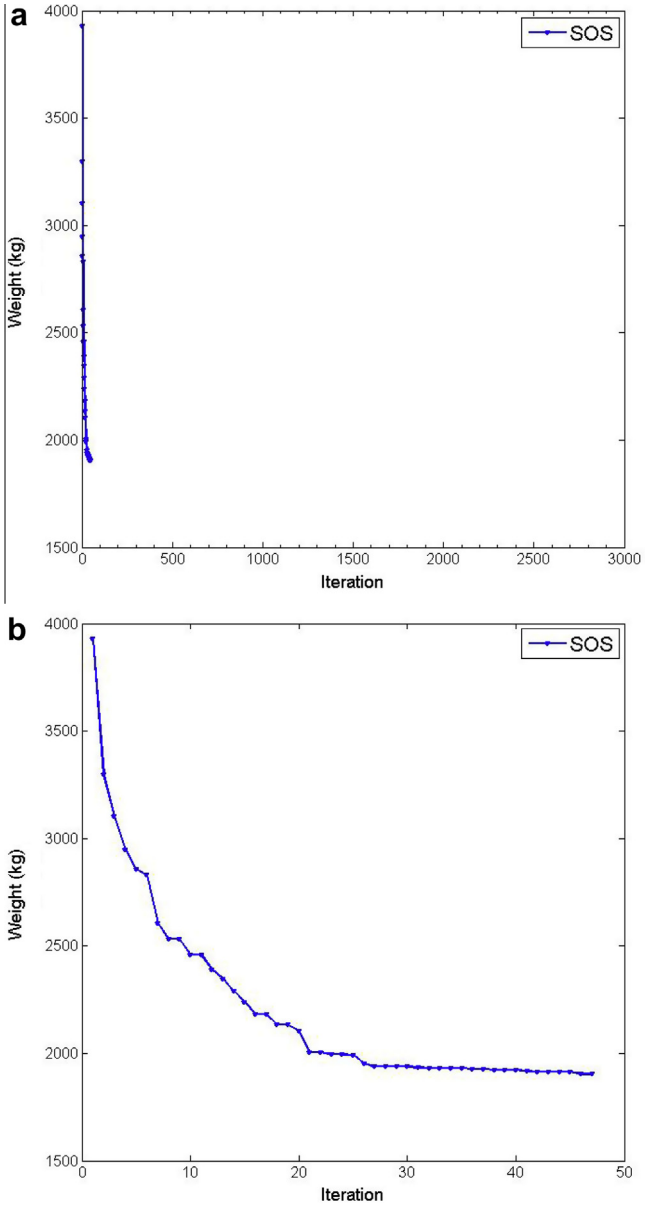

Fig. 16. Convergence rates for the 52-bar truss using the proposed method SOS for: (a) 3000 iterations and (b) 50 iterations.

SOS uses three interaction strategies, mutualism, commensalism, and parasitism, to gradually improve candidate solutions.

- Mutualism is an SOS strategy not used in either PSO or DE. Mutualism modifies candidate solutions by calculating the difference between the best solution and the average of two organisms (Mutual_Vector). Mutual_Vector often produces a unique characteristic, especially when both organisms are located far from one another in the search space, which gives it an advantage in exploring new regions. Further, the two interacting individuals are updated concurrently rather than singly or separately.
- Commensalism is a strategy used by SOS as well as PSO, DE, and CS that alters a solution by calculating the difference between other solutions. Commensalism in SOS differs slightly from that in other algorithms because it uses the best solution as the reference point to exploit promising regions near the best solution. This helps increase search process convergence speed.
- Parasitism is a mutation operator unique to SOS. The trial mutation vector (Parasite_Vector) competes against other randomly selected individuals rather than its parent or creator. The following three items highlight the advantages of parasitism: (1) Parasite_Vector is created by changing the solution of the host in random dimensions with a random number rather than by changing only a small part of the solution. These selected dimensions may vary from one dimension to the entire dimension set. (2) A small number of changed dimensions represents a local search characteristic, while changes to whole dimensions can generate solutions that add a perturbation to the ecosystem that maintains diversity and prevents premature convergence. (3) As a mutation operator, parasitism produces unique solutions that may be located in completely different regions due to the highly random nature of this operator.

- SOS uses greedy selection at the end of each phase to select whether to retain the old or modified solution in the ecosystem. DE also uses this selection mechanism.
- SOS uses only the two parameters of maximum evaluation number and population size. Other algorithms such as GA, DE, PSO, MBA, and CS require the tuning of at least one specific-algorithm parameter in addition to these two parameters. While simpler and more robust than competing algorithms, SOS is able to resolve a wide variety of problems. Moreover, it avoids the risk of compromised performance due to improper parameter tuning.

5. Conclusion

This paper presents a new metaheuristic algorithm called Symbiotic Organisms Search (SOS) inspired by the biological interactions between organisms in an ecosystem. SOS simulated this natural pattern using the three strategies of mutualism, commensalism, and parasitism. Its application to sample problems demonstrated the ability of SOS to generate solutions at a quality significantly better than other metaheuristic algorithms. Based on mathematical benchmark function results, SOS precisely identified 22 of 26 benchmark function solutions, surpassing the performance of GA, DE, BA, PSO, and PBA. SOS was also tested with four practical structural design problems. Results demonstrated SOS was able to achieve better results with fewer evaluation functions than algorithms tested in previous works.

The three phases of the SOS algorithm are simple to operate, with only simple mathematical operations to code. Further, unlike competing algorithms, SOS does not use tuning parameters, which enhances performance stability. We thus conclude that the novel SOS algorithm, while robust and easy to implement, is able to solve various numerical optimization problems despite using fewer control parameters than competing algorithms.

References

[1] Lee KS, Geem ZW. A new structural optimization method based on the harmony search algorithm. Comput Struct 2004;82:781–98.
[2] Osman IH, Laporte G. Metaheuristics: a bibliography. Ann Oper Res 1996;63:513–623.
[3] Holland JH. Adaptation in natural and artificial systems. University of Michigan Press; 1975.
[4] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. Perth, Australia; 1995. p. 1942–1948.
[5] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341–59.
[6] Dorigo M, Stützle T. Ant colony optimization. Bradford Company; 2004.
[7] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60–8.
[8] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71.
[9] Pham DT, Ghanbarzadeh A, Koc E, Otri S, Rahim S, Zaidi M. The Bees algorithm, a novel tool for complex optimisation problems. In: Proceedings of the 2nd international virtual conference on intelligent production machines and systems (IPROMS 2006). Elsevier: Oxford; 2006. p. 454–9.
[10] Yang X-S. Firefly algorithms for multimodal optimization. In: Proceedings of the 5th international conference on stochastic algorithms: foundations and applications. Sapporo, Japan: Springer-Verlag,; 2009. p. 169–78.
[11] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mech 2010;213:267–89.
[12] Kaveh A, Talatahari S. Optimal design of skeletal structures via the charged system search algorithm. Struct Multi Optim 2010;41:893–911.
[13] Erol OK, Eksin I. A new optimization method: big bang–big crunch. Adv Eng Software 2006;37:106–11.
[14] Yang X-S, Deb S. Cuckoo search via levy flights. In: Proceedings of the world congress on nature & biologically inspired computing (NaBIC-2009). Coimbatore, India; 2009. p. 210–214.
[15] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm for optimization of truss structures with discrete variables. Comput Struct 2012;102:49–63.
[16] Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm – a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 2012;110–111:151–66.
[17] Kaveh A, Farhoudi N. A new optimization method: dolphin echolocation. Adv Eng Software 2013;59:53–70.
[18] Kaveh A, Khayatazad M. Ray optimization for size and shape optimization of truss structures. Comput Struct 2013;117:82–94.
[19] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Inf Sci 2013;237:82–117.
[20] Ishibuchi H, Yoshida T, Murata T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 2003;7:204–23.
[21] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1:67–82.
[22] Sapp J. Evolution by association: a history of symbiosis: a history of symbiosis. New York: Oxford University Press; 1994.
[23] Whitley D, Rana S, Dzubera J, Mathias KE. Evaluating evolutionary algorithms. Artif Intell 1996;85:245–76.
[24] Cheng M-Y, Lien L-C. Hybrid artificial intelligence-based PBA for benchmark functions and facility layout design optimization. J Comput Civil Eng 2012;26:612–24.
[25] Deb K. An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 2000;186:311–38.
[26] Chickermane H, Gea HC. Structural optimization using a new local approximation method. Int J Numer Methods Eng 1996;39:829–46.
[27] Gandomi AH, Yang X-S, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 2011:1–19.
[28] Gold S, Krishnamurty S. Trade-offs in robust engineering design. In: Proceedings of the 1997 ASME design engineering technical conferences. Sacramento; 1997.
[29] Wang G. Adaptive response surface method using inherited Latin hypercube design points. J Mech Des 2003;125:210–20.
[30] Zhang YN, Liu JP, Liu B, Zhu CY, Li Y. Application of improved hybrid genetic algorithm to optimize. J South China Univ Technol 2003;33:69–72.
[31] Li LJ, Huang ZB, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables. Comput Struct 2009;87:435–43.
[32] Wu S-J, Chow P-T. Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 1995;56:979–91.
[33] Lee KS, Geem ZW, Lee S-H, Bae K-W. The harmony search heuristic algorithm for discrete structural optimization. Eng Optim 2005;37:663–84.
[34] Kaveh A, Talatahari S. A particle swarm ant colony optimization for truss structures with discrete variables. J Constr Steel Res 2009;65:1558–68.