Knowledge-Based Systems 232 (2021) 107486

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A Novel Smell Agent Optimization (SAO): An extensive CEC study and engineering application

Ahmed T. Salawudeen [a,*], Muhammed B. Mu'azu [b], Yusuf A. Sha'aban [b,c], Adewale E. Adedokun [b]

[a] Department of Electrical and Electronics Engineering, University of Jos, Nigeria
[b] Department of Computer Engineering, Ahmadu Bello University, Zaria, Nigeria
[c] Department of Electrical and Electronics Engineering, University of Hafr Al Batin, Saudi Arabia

ARTICLE INFO

Article history:
Received 4 February 2021
Received in revised form 24 August 2021
Accepted 7 September 2021
Available online 15 September 2021

Keywords:
Smell Agent Optimization
Hybrid renewable energy sizing
Total annual cos
CEC benchmark functions
Statistical analysis

ABSTRACT

This paper presents an extensive study of a new metaheuristics algorithm called Smell Agent Optimization (SAO) on some CEC numerical optimization benchmark functions and Hybrid Renewable Energy System (HRES) engineering problems. The SAO implements the relationships between a smell agent and an object evaporating a smell molecule. These relationships are modelled into three separate modes called the sniffing, trailing and random modes. The sniffing mode simulates the smell perception capability of the agent as the smell molecules diffuse from a smell source towards the agent. The trailing mode simulates the capability of the agent to track the part of the smell molecules until its source is identified. Whereas, the random mode is a strategy employed by the agent to avoid getting stuck in local minima. Thirty-seven commonly used CEC benchmark functions, and HRES engineering problem are tested, and results are compared with six other metaheuristics methods. Experimental results revealed that the SAO can find the global optimum in 76% of the benchmark functions. Similarly, statistical results showed that the SAO also obtained the most cost effective HRES design compared to the benchmarked algorithms.

1. Introduction

Combinatorial optimization is often used to tackle the problem of ordering and selection of discrete objects. The complexity of combinatorial optimization problems increases with increasing dimensionality, leading to very large intractable problems. Obtaining a combination of discrete solutions, which may not necessarily be optimum, is time-consuming and requires high computational resources. The analytical methods of solving such problems require the computation of the gradient to improve successive solutions. These gradients, obtained through linear approximations around certain operating points, may fail when the system moves to a new operation region. Though, gradient-based methods have been enormously applied on small-scale optimization problems, the non-linearity involved in large-scale problems are more than what gradient-based method can handle. These challenges associated with analytical methods have inspired researchers to develop metaheuristics optimization techniques.

Metaheuristics optimization techniques are developed using the intelligent behaviours of various natural systems. In the last few years, these methods have emerged as popular and powerful tools for solving complex engineering optimization problems. The principle of these methods is different from the traditional optimization methods as it requires only the function values, and not the derivatives or gradient information of the problem [1–3]. These function values are at first generated using a random operator to explore different solution spaces, this is called exploration (diversification). Then, the algorithm thoroughly searches these solution spaces for the optimal solution by exploitation (intensification). Balancing between this exploration (diversification) and exploitation (intensification) is a major challenge which must be carefully considered to avoid trapping in local optimal problems. The most common approach to address this problem is through dynamic parameter selection or by hybridizing the positive features of two or more algorithms. Recently, researchers have also considered this imbalance in developing new metaheuristics algorithms [4]. However, majority of these new metaheuristics algorithm were focused only on the foraging behaviours of the

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: atsalawudeen@unijos.edu.ng (A.T. Salawudeen), mbmuazu@abu.edu.ng (M.B. Mu'azu), shaaban@uhb.edu.sa (Y.A. Sha'aban), wale@abu.edu.ng (A.E. Adedokun).

https://doi.org/10.1016/j.knosys.2021.107486
0950-7051/

Nomenclature

Ppv	PV modules output power (W)
TAC	Total annual cost ($)
ACC	Total annual capital cost ($)
AMC	Total annual capital cost ($)
a_r	Interest rate (5%)
P_{Bat}	Price of battery (170$)
C_{Bat}	Present worth of batter
C_{Inv}	Inverter/Converter present worth
P_{Inv}	Inverter/Converter Price (2000$)
T	Temperature of smell molecules
m	Mass of smell molecules
n	Life span of hybrid system (25 years)
n_{pv}	Numbers of PV modules
n_{wt}	Numbers of wind turbine
n_{Bat}	Numbers of batteries
n_{pv_max}	Maximum number of PV modules (350)
n_{pv_min}	Minimum number of PV modules (1)
n_{wt_max}	Maximum number of wind turbines (350)
n_{wt_min}	Minimum number of wind turbines (1)
n_{Bat_max}	Maximum number of battery (350)
n_{Bat_min}	Minimum number of battery (1)
LCE	Levelized cost of energy ($/kWh)
C_{PV_m}	Maintenance of cost of PV modules ($)
$E_{e/d}(t)$	Excess/Deficit Energy (kW)
SOC_{max}	Maximum state of charge (2.4 Wh)
SOC	Battery state of charge
DOD	Dept of discharge (0.8%)
ω	Hourly self-discharge rate (0.0002 Wh)
S_{bat}	Nominal capacity of battery (2.4)
η_{BC}	Battery bank discharge efficiency (85%)
η_{Inv}	Inverter efficiency (95%)
Cpv	Unit cost of PV (240$)
Cwt	Unit cost of battery (1250$)
Cpv_m	Maintenance cost of PV modules (15$)
Cwt_m	Maintenance cost of wind turbine (30$)
I(t)	Solar Insolation (kW/m^2)
A_{PV}	Area of PV module (2 m^2)
η_{PV}	Efficiency of PV modules (15.7%)
v(t)	Instantaneous wind speed (m/s)
v_{ci}	Cut-in wind speed (2 m/s)
v_{co}	Cut-out wind speed (25 m/s)
v_r	Rated wind speed (9 m/s)
P_r	Rated power of wind turbine (1 kW)
LSP	Loss of Power Supply (kW)
LPSP	Loss of Power Probability (%)
P_{Load}	Total load demand (kW)
C_{wt_m}	Maintenance of cost of wind turbines ($)
P_{Gen}	Power generated (kW)
SL	Step Lenth
SOC_{min}	Minimum state of charge (Wh)

agents as they seek to identify the location of food within their environment [5,6]. The nature and concentration of food source which have a beneficial influence on the behaviours of the agent seldom considered, except in a few cases [3]. For example, the movement of fish in the water is attracted towards the region of high concentration of food source(s). This attraction may restrict the fish from searching other possible areas, thus, reducing the exhaustive exploration of the search space. Similarly, considering only the foraging movement of the agents towards the food source does not truly represent the real-life scenario of a biological system. This contributes to the problem of imbalance between diversification and intensification, increasing the chances of poor convergence in most of the algorithms and may subsequently lead to local optima problems [7]. It is also important to stress that, there is no known single modern optimization method that can solve all real-world optimization problems effectively, "The No-Free Lunch Theorem". Especially, since all real-world problems have their peculiar characteristic which can be likened to the behaviours of a particular optimization algorithm. For instance, the behaviour of ant systems towards pheromone deposit can be described using lane and edge detection problems in image processing. In the same vein, the trajectory or path planning problem in robotics can also be described as an agent trailing a smell molecule to identify its source. The smell molecules diffuse through the air and travel into the olfactory organ (nose) of a smell agent where it interferes with some specialized cell (receptor) that send message to the brain for adequate interpretation. Considering these interactions between an agent with its surrounding environment and bearing in mind the "No Free Lunch Theorem", this paper presents an extensive study of a new metaheuristics' algorithm called the smell agent optimization. The smell agent optimization simulates the interaction between an agent and its surrounding environment into three distinctive modes called the sniffing, trailing and random modes. The sniffing mode mimics the smell perception capability of agents as the smell molecules evaporate from the smell towards the agent. The trailing mode simulate the behaviour of the agent towards identifying the smell source whereas the random mode helps the agent to maintain an optimum path in case of a loss of trail.

The rest of this paper is structured as follows: Section 2 presents a comprehensive review of related works; the SAO algorithm is presented in Section 3; Section four provides the methods of evaluation on a total of 37 commonly used benchmark functions and hybrid renewable energy systems design; The performance of SAO is compared with some well known and state of the art algorithms in Section 5; The paper is concluded in Section 6, with some recommendations for further work.

2. Related works

This section presents a critical review of some selected popular and state of the art metaheuristic algorithms. These selections were made considering a wide range of algorithm inspired by various natural phenomenon. The underlying mechanisms (how the agent identifies the "food source" and the influence of the "food source" on the behaviour of the agent), strengths and weakness are highlighted.

Particle Swarm Optimization (PSO) is one of the most widely accepted metaheuristic algorithms. The idea of PSO is inspired by the intelligent behaviours of school of fishes and flocks of birds [8]. Particles in PSO are initialized by a set of randomly generated initial positions. Each member particles in the swarm are assigned a random velocity and the particles (potential solutions) are flown through the optimization hyperspace. Particles in the swarm keeps track of its own coordinate in the hyperspace which is affiliated with the best fitness (solution) it has achieved so far (personal best *"pbest"*). The particles also collectively keep track of the overall best fitness (solution) and its location in the swarm (global best *"gbest"*). At each time step in the PSO, a particle changes its velocity (or acceleration) as it moves towards

the *"pbest"* and *"gbest"* values until the particle with the best fitness is found. The performance of the PSO algorithm over several engineering problems has demonstrated its effectiveness in solving several optimization problems effectively. However, the original PSO has been allied with local optimal problems due the imbalance between intensification and diversification. This is usually attributed to the lack of guiding (control) parameter(s) as the algorithm swarm through the optimization hyperspace. Over the years, several researchers have proposed modifications to the PSO with a common goal of improving its precision and rate of convergence. For example, ref [9] developed an improved PSO for solving non-convex economic load dispatch problem. In this work, a Gaussian random variable is introduced in the generation of particles velocity. This is with the hope of improving the searching efficiency and ensuring a high probability of obtaining the global optimum without hindering the acceleration convergence and increasing the complexity of the structure of PSO. Also, ref [10] presented a modified PSO for time dependent vehicle routing problem (TDVRP) with the aim of minimizing travel time and reducing carbon emission. Crossover and mutation operators were introduced in order to increase the diversity of the original PSO by replacing half of the worst individual with a newly generated particle. In the work of Delice, et al. [11] a modified PSO algorithm with negative knowledge is proposed for solving the mixed-model two sided assembly line balancing problem. A combine selection and decoding procedure was employed at the generation stage of the PSO. This enhanced the exploration capability of the original PSO and also allowed the PSO to search at different points in the hyperspace.

Artificial Fish Swarm Algorithm (AFSA) is inspired by the foraging behaviour of fish in water. This behaviour is classified into preying, swarming and chasing [12]. In each classification, three communication rules (synergetic rule, reconnaissance rule and stochastic rule) are employed to decide the movement of each fish. The algorithm employs control parameter such as *visual* which is a synonymous to how far is the food source from the location of the fish, *step* which penalize the amount of movement the fish can take to get to the food source and crowd factor, which determine the concentration of other fishes around the food source to guide its evolution. At the initial stage of AFSA, the population of the artificial fish is generated randomly and are placed in the hyperspace for foraging. Then, the preying behaviour is implemented. If the solution obtained during preying is not better than that obtained during foraging then, the swarming behaviour is implemented. Also, the solution obtained during swarming is not better than that obtained during praying, the chasing behaviour is implemented. At the end, the behaviour with the best performance is selected. The performance of the algorithm over several optimization problems demonstrated the efficiency and capability of the AFSA. Just like in most other metaheuristic algorithm, the major challenge of the AFSA is the ease at which the algorithm falls into local optimum. This is attributed to the constant effect of the control parameters when the algorithm evolved from generation to generation. To address these challenges, several researchers have proposed modifications to the standard AFSA. In [13], an improved cultural AFSA (with four variants) called the weighted AFSA (wCAFAC) was developed using the normative and situational knowledge inherent in Cultural Algorithm (CA) to increase the diversification of the algorithm. An inertial weight was introduced such that the control parameters could be selected adaptively and a crossover operator was introduced such that, an offspring could inherit some of the characteristics of its parents. Simulation results on standard benchmark functions showed a significant improvement over the original AFSA. The work presented in Luan, et al. [14] introduced a three parameter Lorentzian function into the AFSA with the hope of improving its visual distance through adaptive selection. At the initial stage of the AFSA, a large visual distance is selected to enhance both the searching capability and convergence of the algorithm while at the later stage of AFSA, a small visual distance is adopted to improve the local searching ability and the accuracy of the optimal solution. Also, to keep a balance between the iteration speed and precision, a variable step adaptive operator is introduced. Ref. [15] presented a modified AFSA for fuzzy time series forecasting in which, the chemotaxis behaviour of bacterial foraging was employed in the foraging phase of the AFSA. Thereafter, the Levy flight was introduced as a mutation operator for a mutation strategy. Simulation results also demonstrated the efficiency of this approach.

The Artificial Bee Colony (ABC) which is one of the popular algorithm was developed using the swarming behaviours of wild honey bees [16,17]. To explore a food source, a forager bee evaluates its closeness to the hive, its richness, taste of its nectar and the ease of extracting the energy. This information is employed to model the ABC algorithm consisting of three distinctive forms; employed bees, onlooker bees and the scout bees. The employed bee searchers for food sources and pass the information to the onlooker which then chooses the best location of food source to be exploited. Scout bee spontaneously searches for new location of food source after the exhaustive exploitation of already known food source have been carried out. Although the ABC algorithm have been enormously applied to solve various optimization problems, several researchers have proposed modifications to improve its overall performance. In ref [18], an improved ABC algorithm was proposed using the explosion search mechanism of fireworks algorithm. The stages in the improved ABC were classified into searching stage which consists of the three searching strategies of the original ABC and fireworks explosion stage. The fireworks explosion stage is used to exploit the promising region from the first stage, with the hope to find a better solution. Ref. [19], implemented an ABC with adaptive covariance matrix. The distribution information retrieved from the population of ABC is used to establish a proper covariance-based coordinates. The search operators of the ABC were implemented on eigen and natural coordinates. Experimental analysis proved the superiority of the covariance-based ABC. In the same vein, other popular algorithm that have demonstrated competitive performance over time includes: The Ant Colony Optimization (ACO) which was inspired by the pheromone laying and trailing behaviours of natural ants as a medium of communication and survival [20]. Cuckoo Search (CS) algorithm, inspired by the obligate brood parasitism of certain cuckoo species [21], Bacterial Foraging Optimization (BFO) algorithm, inspired by the foraging behaviours, chemotaxis, swarming, reproduction and elimination and dispersal of E. coli bacterial [22]. Firefly Algorithm (FA) inspired by the flashing behaviours of fireflies towards attracting each other [23]. Bat Algorithm (BA) inspired by mimicking the natural pulse loudness and emission rate of natural bats [24], etc.

In the last few years, several metaheuristic algorithms have been developed with distinct mechanism to balance between exploration and exploitation. In [25] for example, a novel nature inspired optimization algorithm called Smell Detection Agent (SDA) was developed. The principle of the SDA involved the creation of an artificial surface with smell trails and the use of the dog in resolving a path around this surface. The SDA was focused on the ability of a dog to lock onto a scent using their highly developed olfactory cells and their territorial behaviours through urination. The dogs in the SDA hold two parameters, namely the signature value which is used by the dogs to mark smell spots and the radius value, which indicates the smelling ability of the dogs. In the SDA, the search environment is initialized with a randomly selected smell spot and smell values are assigned to

be inversely related to the Cartesian distance between the source and the destination. At this point, the agents are initialized to explore each spot. The smell spot that has the highest value is chosen as the unmarked point during iteration and the SDA is moved to that spot, this helps the algorithm during exploitation. Ref. [26], developed a new optimization algorithm mimicking the animal behavioural ecology called the Optimal Foraging Algorithm (OFA). The OFA position is expressed as a two-dimensional variable of longitude and latitude and the foragers can move along the two directions. When a forager searches the location with the abundance of food, these foragers pass the information to others through recruitment. The information sharing process continue until the entire food source is exploited. The Crow Search Algorithm (CSA) was inspired by the idea that, Crows mostly hide the excess food in hidden places to feed on them when the need arises [27]. Ref. [28] presented a new metaheuristic algorithm called the Pathfinder Algorithm (PFA). The idea of PFA was based on the collective movement of swarm of animals and the leadership hierarchy employed to find the best location for food to pray. Salp Swarm Algorithm (SSA) was developed to mimic the swarming behaviours of salps in nature [29]. In SSA, the initial population is divided into leader and followers. The salps which has the best initial position are assumed as the leader salp while all other salps are followers. The leader salp coordinate and direct the activities of the order salp all through the searching process. Recently, an algorithm inspired from the concept of distribution of arithmetic operators such as addition, multiplication, subtraction and division was developed into Arithmetic Optimization Algorithm (AOA) [4]. The exploration phase of AOA was implemented using the multiplication and division operators of basic arithmetic. This is due to the high computational characteristics of division and multiplication as compared to addition and subtraction. On the other hand, the addition and subtraction operators are used to model the exploitation phase of the AOA. Also, recently, an algorithm which was inspired by the intelligent behaviours of aquila birds was developed in [30]. The operational principle of the optimizer called Aquila Optimizer (AO) is represented in for main methods. First method is selecting search space by higher soar with vertical stoop. Second method is search space exploration by contour flight with short glide attack. The third is exploiting around a converged search space by low flight with slow descent attack and lastly, swooping by walk and grab prey. For detail review of other solution methods consider the following references [31–37]. Brief summary of some of the popular metaheuristics' algorithms are given in Table 1.

In line with the discussion thus far, this paper presents an extensive study of a new metaheuristic algorithm developed using the interaction between a smell agent, smell molecules and a smell source. Similar to SDA, the new algorithm is inspired by the concept of smell perception. However, unlike in SDA where smell molecule is considered as substances deposited in a rectangular Cartesian coordinate, the smell in the new algorithm is regarded as gaseous molecules evaporating from a source. In SDA the agent is the dog which marks its territory by urinating and identifying an established smell spot. In contrast, the agent in the new algorithm is any biological agent that has the ability of olfaction. The new algorithm called Smell Agent Optimization (SAO) has three distinctive behaviours; sniffing, trailing and random modes. The sniffing mode is based on the Brownian movement of smell molecules. The trailing mode is based on the olfaction capability of an agent and its ability to follow the smell molecules to its source. The random mode is the strategy the agent employs to escape from being trapped in a local optimum as a result of loss of the smell trail. The agent in SDA is passive and has no active participation in the computational process [54]. In contrast, the agent in SAO is active and plays a significant role in the optimization process. We experiment the SAO with six other metaheuristics algorithms on thirty-seven commonly used CEC benchmark functions. To further justify the acceptability of SAO, we employed the algorithm to solve the real-world optimization problem of off-grid hybrid renewable power systems design. The contributions of this paper are highlighted as follows:

i. An extensive experimental study of a new metaheuristics algorithm called Smell Agent Optimization (SAO) which is based on the interaction between a biological organism that has the ability of olfaction and an object evaporation a small molecule is presented. This experiment was done using 37 CEC benchmark functions.
ii. Detailed comparative study of SAO was done using some well-established algorithms in the form of ABC, CSA, GA, PSO, OFA and AOA is presented using statistical metrics.
iii. The SAO is used to design an HRES engineering problem considering three independent configurations namely; PV/Wind/Battery, PV/Battery and Wind/Battery. An extensive results discussion and comparative analysis were done with the five algorithms listed in ii.
iv. We analysed the economic viability of our HRES models using Loss of Power Supply Probability (LPSP), Levelized Cost of Energy (LCE) and Excess/Deficit Energy. This economic analysis is hardly considered in most of the previous studies.

3. Smell Agent Optimization (SAO) algorithm development

The sense of smell is one of the primary senses through which the world is perceived. Most living organisms perceive the presence of harmful chemical substances in their environment using the sense of smell [55–57]. It seems natural to consider human olfaction in developing SAO. However, we discovered that most biological agent use olfaction for the same primary purpose, searching for food, mating and avoiding danger [57–59]. This motivated the development of a generalized smell agent-based optimization algorithm.

3.1. Sense of smell (olfaction)

The term smell refers to the power to perceive (sense) the odour of a substance through olfactory nerves [60]. The smell contains molecules of substances diffusing from a source in a non-uniform manner with a molecular weight of less than 300 Dalton [60,61]. Detecting and discriminating these smell molecules are critical for the survival of most agents [59]. The process of smell perception is similar amongst agents that have the capacity of olfaction [57,60–62]. An essential element in distal odours identification is the ability to trail odour plume. This ability is very critical to the existence of several organisms including the human beings [61]. With odour perception, humans can distinguish harmful substances such as: iron pills, cleaning products, button batteries, carbon monoxide, pesticides, wild mushrooms and hydrocarbons. Detail information on smell perception can be found in [2,54,55,57,60,63].

3.2. SAO algorithm

Three distinct modes govern the general framework of the SAO. The idea of these modes is derived from the smell perception steps mentioned above. In the first mode, the agent perceives the smell molecules, evaluates its position and decide whether to search for its source or not. In the second mode, the agent trails the smell molecules in the search for the smell source base on the decision from the first mode. The third mode helps the agent to

Table 1
Summary of some popular optimization algorithms.

Algorithm	Inspiration	Ref.
Particle Swarm Optimization – PSO	Inspired by school of fishes or flocks of birds	[8]
Genetic Algorithm – GA	Inspired by the theory of genetic evolution	[38]
Differential Evolution – DE	Inspired by the theory of natural theory of evolution	[39]
Artificial Fish Swarm Algorithm – AFSA	Inspired by the foraging behaviours of fishes in water	[12]
Artificial Bee Colony – ABC	Inspired by the swarming behaviour of honey bees in nature	[19]
Cultural Algorithm – CA	Inspired by the principles of cultural evolution processes	[40]
Ant Colony Optimization – ACO	Inspired by the pheromone railing behaviours of natural ants	[20]
Sine Cosine Algorithm – CSO	Inspired by the concept of trigonometric sine and cosine functions	[41]
Black Hole Algorithm – BHA	Inspired by simulating the behaviours of black hole in outer space	[42]
Bacterial Foraging Optimization – BFO	Inspired by the foraging behaviours of E. coli. Bacteria	[22]
Firefly Algorithm – FA	Inspired by the flashing behaviours of natural fireflies	[23]
Bat Algorithm – BA	Inspired by the natural pulse loudness and emission rate of natural bats	[24]
Optimal Foraging Algorithm – OFA	Inspired by the principle of animal behavioural ecology	[26]
Crow Search Algorithm – CSA	Inspired by the intelligence behaviours of crows in hidden food items	[27]
Ant Lion Optimizer – ALO	Inspired by the interaction between ants and ant lions in nature	[43]
Whale Optimization Algorithm – WOA	Inspired by the humpback hunting behaviour of whale species	[44]
Grey Wolf Optimization – GWO	Inspired by the hunting strategies and leadership hierarchy of grey wolf	[45]
Invasive Weed Optimization – IWO	Inspired by the survival strategies of weed phenomena	[46]
Cat Swarm Optimization – CSO	Inspired by the tracing and seeking mode behaviours of natural cats	[47]
Moth Flame Optimization – MFO	Inspired by the transverse operation moth–flames towards the moon	[48]
Cuckoo Search – CS	Inspired by the aggressive reproduction strategies of cuckoo species	[21]
Artificial Jellyfish Search – AJS	Inspired by the behaviours of jellyfish in following the ocean current	[49]
Symbiotic Organisms Search – SOS	Inspired by the symbiotic interaction of organism in ecosystem	[50]
Path Finder Algorithm – PFA	Inspired by the hierarchical behaviours of swarm of organism	[28]
Salp Swarm Algorithm – SSA	Inspired by the swarm behaviours of natural salps in water.	[29]
Arithmetic Optimization Algorithm – AOA	Inspired by the concept of arithmetic operators	[4]
Aquila Optimizer – AO	Inspired by the foraging behaviours of aquila birds	[30]
Atomic Orbital Search – AOS	Inspired by the principles of quantum based atomic model	[51]
Solar System Algorithm – SSA	Inspired by the orbiting behaviours of objects around solar systems	[52]
Adolescent Identity Search Algorithm – AISA	Inspired by the principle of identity developed of adolescent peer groups	[53]

avoid getting trapped in a local minimal should in case the agent loss its trail. This description is highlighted in algorithms 1.

Algorithm 1: SAO Description

Input: X: $n \times m$ matrix of smell molecules, other algorithm parameters;

Output: Y : $n \times 1$, position of best smell molecules

$n \leftarrow$ size(X)

$best \leftarrow$ an empty dictionary

for $k = 1$ to n do

Sniffing

Trailing

Random

y: #return solution of best mode

end for

$best_{sorted} \leftarrow$ **sort**(y)

$y \leftarrow$ key$(best_{sorted}[0])$

3.2.1. Sniffing mode

Since smell molecules diffuse in the direction of the agent, we first initiate the process with a randomly generated initial position of smell molecules. Let the total number of smell molecules be N and the total variables in the hyperspace (i.e. number of decision variables) be denoted as D, then, the smell molecules can be initialize using Eq. (1):

$$x_i^{(t)} = \begin{bmatrix} x_{(1,1)} & x_{(1,2)} & x_{(1,D)} \\ \vdots & \vdots & \vdots \\ x_{(N,1)} & x_{(N,2)} & x_{(N,D)} \end{bmatrix} \tag{1}$$

The position vector in Eq. (1) enables the agent to determine its own best position in the search space and can be generated using Eq. (2)

$$x_i^{(t)} = lb_i + r_0 \times (ub_i - lb_i) \tag{2}$$

where ub and lb are the upper and lower bound defined for the decision variables respectively and r_0 is a random number generated in the range of (0,1].

In line with Eq. (1) each smell molecules is assigned an initial velocity through which they diffuse from the smell source/origin, using Eq. (3)

$$v_i^{(t)} = \begin{bmatrix} v_{(1,1)} & v_{(1,2)} & v_{(1,D)} \\ \vdots & \vdots & \vdots \\ v_{(N,1)} & v_{(N,2)} & v_{(N,D)} \end{bmatrix} \tag{3}$$

In the geometric number space, every smell molecule represents a candidate solution. The positions of these candidate solutions (smell molecule) can be determined from the position vector $x_i^t \in \Re^N$ given in Eq. (1) and the molecules' velocity $v_i^t \in \Re^N$ given in Eq. (3). Since the smell molecules diffuse in Brownian form, the velocity of the molecules is updated using Eq (4).

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \times \Delta t \tag{4}$$

In the SAO, Δt is assumed to be 1, indicating that the agent takes a progressive step one at a time during the optimization process. For example, the algorithms increase its iteration progressively by 1 until the final iteration is reached. Therefore, the new position of the smell molecules is Eq. (5)

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{5}$$

Every smell molecule has its corresponding diffusion velocities for which they evaporate and update their position in the search space. Since the smell molecules diffuse in a nonuniform manner until it reaches the location of the agent, the update velocity of the smell molecules is computed using Eq. (6)

$$v_i^{(t+1)} = v_i^{(t)} + \upsilon \tag{6}$$

where υ is the update component of the velocity given by Eq. (7)

$$\upsilon = r_1 \times \sqrt{\frac{3kT}{m}} \tag{7}$$

The symbol k is the smell constant which normalizes the effect of temperature and mass on the kinetic energy of the smell molecules, T and m are temperature and mass of the smell molecules, respectively.

Both temperature (T) and mass (m) are parameters associated with smell molecules initialization, thus, has no direct effect on the stability of the algorithm. The idea of T and m is derived from the ideal theory of gas. For example, since a smell molecule is Brownian in nature, like gas molecules, its velocity update in Eq. (7) is derived from the concept of the hydro static pressure of gases, given as [64,65]:

$$\frac{1}{2}nm\upsilon^2 = \frac{3}{2}knT \tag{8}$$

Thus, Eq. (7) was derived by making the velocity component (υ) in Eq. (8) the subject. To avoid unnecessary waste of time in selecting the values of m and T, their values has been experimentally determined to be 0.175 and 0.825 respectively.

The fitness of the updated positions of the smell molecule in Eq. (5) is evaluated. At this point, the sniffing mode is said to have been completed and the position of the agent x^t_{agent} can be determined (position of best fitness).

3.2.2. Trailing mode

This mode simulates the searching behaviours of an agent towards identifying the source of a smell. While searching for smell source, the agent may sniff a new position with a higher concentration of smell molecules than its present position, the agent moves towards this new position using Eq. (9)

$$x_i^{(t+1)} = x_i^t + r_2 \times olf \times (x_{agent}^{(t)} - x_i^{(t)}) - r_3 \times olf \times (x_{worst}^{(t)} - x_i^{(t)}) \tag{9}$$

where, r_2 and r_3 are random numbers in the range of (0,1]. The r_2 penalizes the influence of olf on $x_{agent}^{(t)}$ and r_3 penalizes the effect of the olf on $x_{worst}^{(t)}$.

For the agent to efficiently trail the path of the smell, the agent records the fitness of its present position $x_{agent}^{(t)}$ and the position with the worst smell fitness- $x_{worst}^{(t)}$ obtained from the sniffing mode. This information helps the algorithm in improving the balance between exploration and exploitation as shown in Eq. (9). Since olfaction capacity mainly depends on the size of the olfactory lobes, psychological and physical conditions of agents [66], the value of olf should be carefully selected. For example, a small value of olf indicate poor olfaction which favours local searching, whereas a large value of olf indicates stronger olfaction which favours global searching of SAO.

3.2.3. Random mode

The smell molecules are discrete and if they are separated by large distances apart in comparison with the search space, the intensity/concentration of the smell molecule may vary over time. This variation can "bemuse" the agent, leading to the loss of the smell and making trailing a challenge. At this point, the agent may be trapped in local minima due to its inability to continue trailing. If this occurs, the agent goes into the random mode which is described by:

$$x_i^{(t+1)} = x_i^{(t)} + r_4 \times SL \tag{10}$$

where, SL is a constant, indicating the step length and r_4 is a random number which stochastically penalizes the value of SL. In a situation where the trailing mode fails to obtain the best fitness or identify the smell source or the agent loses its trail, the agent takes a random step using Eq. (10).

Fig. 1, shows the conceptual framework of SAO. From the figure, the agent is represented as man and the circle depicts the smell source. Smell molecules are denoted as dotted lines, while the thick line in black represents the path of smell molecules with highest concentration. The x_{fe} are all feasible path leading to the smell source. However, the agent always tries to maintain its position on the path with the highest concentration of smell molecule. At every stage in the searching process, the agent takes note of its position, $x_{agent}^{(t)}$ and the worst positions $x_{worst}^{(t)}$, i.e. the feasible paths, this information is used during trailing as described in Eq. (9). The paths labelled x_{inf} are infeasible paths, which leads the agent into an infeasible solution. However, the agent is always constrained to search within the feasible region by the optimization problem. The detailed implementation of SAO is described in the flowchart given in Fig. 2 and the pseudo-code given in Algorithm 2

Algorithm 2: Pseudo-Code of SAO algorithm

```
Initialize Parameters
Initialize smell molecules initial position
Evaluate fitness
Determine position of agent and worst position of molecules
while (Itr<Itr_max) do:
    for (i=1 to molecules) do:
        for (j=1 to position) do:
            update molecules velocity and position (sniffing)
        end for
        Evaluate fitness
        if (new fitness is better) then:
            update fitness
            update agent and worst molecules
        end if
    end for
    for (i=1 to molecules) do:
        for (j=1: positions) do:
            update position (trailing)
        end for
            Evaluate fitness
    end for
    if (new fitness is better) then
        accept new fitness
        update position
    else
        for (i=1 to molecules) do:
            for (j=1 to position) do:
                Implement random mode
            end for
        end for
    end if
end while
return Optimum solution
```

4. Methods of evaluation

This section is divided into two subsections. The first subsection discuss the procedures employed to evaluate the performance of SAO on 37 CEC benchmark functions as compared with five (5) other metaheuristics algorithms such as; ABC, GA, PSO, CSA and OFA. The second subsection provide a detailed discussion on the application of these algorithms to the sizing of the hybrid renewable energy engineering problem.

4.1. CEC benchmark functions

The CEC benchmark functions are applied mathematical optimization functions used to first check the performance of any

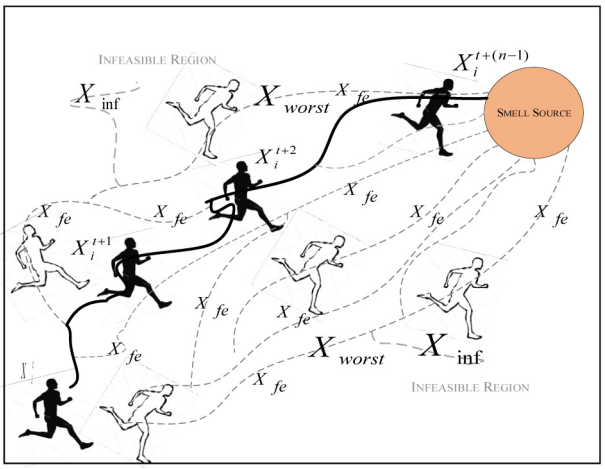

Fig. 1. Conceptual framework of SAO.

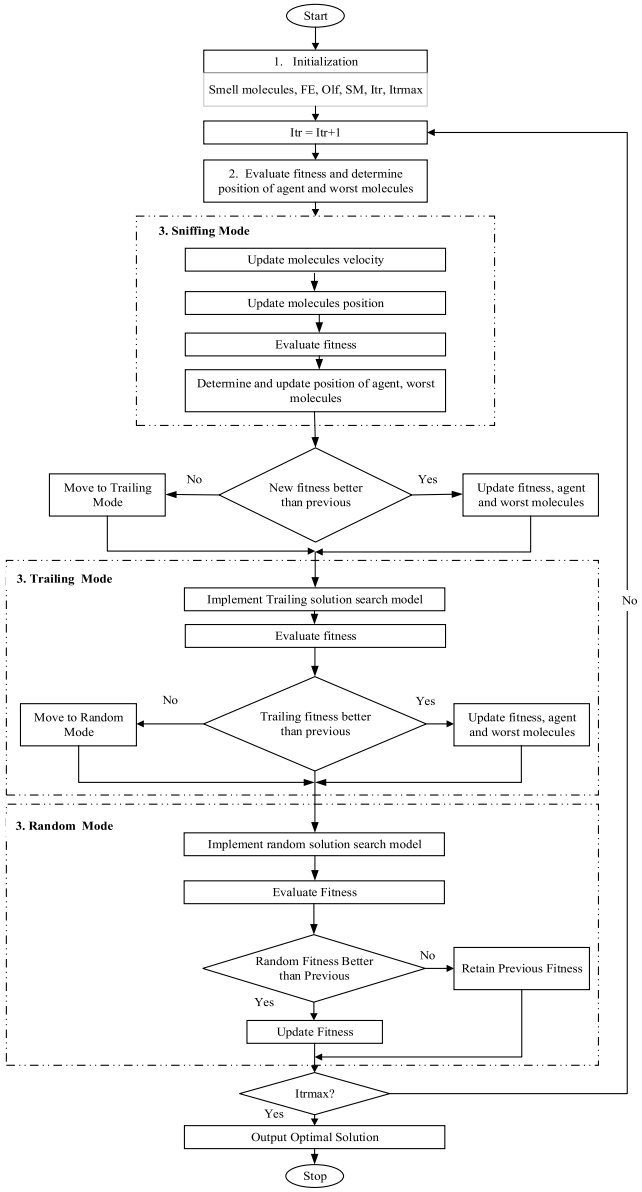

Fig. 2. Flow chart of SAO.

Table 2
Parameter settings of the algorithms.

Sn	Algorithms	Parameters		
1	ABC	*mcn=100*	*Onlocker* = 50	*a=1*
2	CSA	*AP=0.1*	*fl=2*	–
3	GA	*M=0.2*	*C=0.8*	–
4	OFA	$k=Itr/Itr_{max}$	–	–
5	PSO	*w=0.74*	*C1=1.5*	*C2=2*
6	SAO	*olf=0.75*	SL=0.9	–

Where; *mcn* = maximum cycle number, *Onlooker* = onlooker bee, *a* = acceleration coefficient, *c* = crossover rate, *m* = mutation rate, *w* = inertial weight factor, *C1* = local search learning coefficient, C2 = global search learning coefficient, *AP* = awareness probability, *fl* = flight length, *k* = Scale Factor, *Olf* = Olfaction Capacity, SL = Step and Itr_{max} = Maximum Iteration.

newly developed optimization algorithm. In this paper, a total of thirty-seven of these functions with divers' properties were selected. These selected functions were grouped into separate clusters namely unimodal and non separable functions (UN), unimodal and separable functions (US), and multimodal and non separable functions (MS). It should be noted that unimodal functions have a single global optimum which helps to examine the exploitation capability of SAO, while the multimodal functions have two or more local optimums, thus helping to examine the exploration capability of SAO. A clearer depiction of these clusters is given in Tables 3–5. Table 3 holds two-dimensional benchmark functions in the range of F1 to F19, Table 4 holds three-, four-, five- and ten-dimensional benchmark functions in the range of F20 to F27 and Table 5 holds thirty-dimensional benchmark functions in the range of F28 to F37. Readers can visit these papers for further details on each test functions [67–70].

In addition to implementing the SAO algorithm on the stated benchmark functions, we carried out our experiments with five other algorithms for comparison purpose. All the algorithms were simulated using MATLAB R2020b and implemented on HP system running Windows 10 Pro with Intel Core i7 7th Gen CPU @2.70 GHz and 8.0 GHz of RAM. To reduce the effect of parameters on the performance of the algorithms, the same initial population of 50 is used for all the algorithms. The parameter values from the original authors of each algorithm were adopted as shown in Table 2. For all the experiments carried out, we employed function evaluation as stopping criteria with a maximum (maxFE) value of 300,000. To ensure a more stable performance of each algorithm, we carried out our evaluations on 30 independent runs. Four statistical metrics namely best, average, worst and the standard deviation was used to decide which algorithm perform best on any of the benchmark functions. In a situation where none of the algorithms obtains the global solution to any of the benchmark functions, we use a non-parametric statistical (Wilcoxon rank-sum) test, at a 5% (P-value <0.05) level, to show the statistical significance of such result. To analyse the convergence performance of the algorithms, we use a commonly used metrics called the Acceleration Rate (AR) which is calculated as [69].

$$AR = {}^{FE_a}/_{FE_b}$$

where; FE_a is the number of function evaluations of reference algorithm, SAO in this case and FE_b is the number of function evaluation of the other algorithms (ABC, CSA, GA, PSO and OFA). If AR $>$ 1, it implies that the reference algorithm has a faster convergence and when AR $<$ 1, the convergence of the other algorithm is faster.

4.2. Hybrid renewable energy problem formulation

The major challenge with wind and solar energy is their total dependence on weather and climatic conditions [71,72]. The

Table 3
Two dimensions benchmark test functions.

F_{No}	Name	D	Formula	C	Range	Fmin
F1	Adjiman	2	$f(x)=\cos(x_1)\sin(x_2)-\frac{x_1}{(x_2^2+1)}$	NS, MM	[−1,−1;2,1]	−2.0218
F2	Beale	2	$f(x)=(1.5-x_1+x_1x_2)^2+\left(2.25-x_1+x_1x_2^2\right)^2+\left(2.625-x_1+x_1x_2^3\right)^2$	NS, UM	[−4.5, 4.5]	0
F3	Bird	2	$f(x)=\sin(x_1)e^{(1-\cos(x_2))^2}+\cos(x_2)e^{(1-\sin(x_1))^2}+(x_1-x_2)^2$	NS, MM	[−2π, 2π]	−106.765
F4	Bohachevsky1	2	$f(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$	NS, MM	[−100, 100]	0
F5	Booth	2	$f(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	NS, UM	[−10, 10]	0
F6	Branin RCOS1	2	$f(x)=\left(x_2-\frac{5.1x_1^2}{4\pi^2}+\frac{5x_1}{\pi}-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10$	NS, MM	[−5,0; 10,15]	0.3979
F7	Branin RCOS2	2	$f(x)=\left(x_2-\frac{5.1x_1^2}{4\pi^2}+\frac{5x_1}{\pi}-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\times\cos(x_1)\cos(x_2)\ln(x_1^2+x_2^2+1)+10$	NS, MM	[−5;15]	−0.17989
F8	Brent	2	$f(x)=(x_1+10)^2+(x_2+10)^2+e^{-x_1^2-x_2^2}$	NS, UM	[−10 ; 10]	0
F9	Bukin F6	2	$f(x)=100\sqrt{\lvert x_2-0.01x_1^2\rvert}+0.01\lvert x_1+10\rvert$	NS, MM	[−15, −3; −3,3]	0
F10	Camel – Six Hump		$f(x)=(4-2.1x_1^2+\frac{x_1^4}{3})x_1^2+x_1x_2+(4x_2^2-4)x_2^2$	NS, MM	[−5; 5]	−1.0316
F11	Chichinadze	2	$f(x)=x_1^2-12x_1+11+10\cos(\frac{\pi x_1}{2})+8\sin(\frac{5\pi x_1}{2})-(1/5)^{0.5}\exp(-0.5(x_2-0.5)^2)$	S, MM	[−30; 30]	−42.944
F12	Deckkers–Aarts		$f(x)=10^5x_1^2+x_2^2-(x_1^2+x_2^2)^2+10^{-5}(x_1^2+x_2^2)^4$	NS, MM	[−20; 20]	−24 777
F13	Easom	2	$f(x)=-\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$	S, MM	[−100, 100]	−1
F14	Matyas	2	$f(x)=0.26\left(x_1^2+x_2^2\right)-0.48x_1x_2$	NS, UM	[−10, 10]	0
F15	McComick	2	$f(x)=\sin(x_1+x_2)+(x_1-x_2)^2-(3/2)x_1+(5/2)x_2+1$	NS, MU	[−10, 10]	−1.9133
F16	Michalewicz2	2	$f(x)=-\sum_{i=1}^{2}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	NS, MM	[0, π]	−1.8013
F17	Quadratic	2	$f(x)=-3803.84-138.08x_1-232.92x_2+128.08x_1^2+203.64x_2^2+182.25x_1x_2$	NS, MM	[−10, 10]	−3873.72
F18	Schaffer	2	$f(x)=\sum_{i=1}^{30}(x_i^2+x_{i+1}^2)^{0.25}\{[\sin 50(x_i^2+x_{i+1}^2)^{0.1}]^2+1\}$	NS, MM	[−100, 100]	0
F19	Styblinski–Tang	2	$f(x)=\frac{1}{2}\sum_{i=1}^{n}\left(x_i^4-16x_i^2+5x_i\right)$	NS, MM	[−5, 5]	−78.332

Table 4
Three, four, five and ten-dimensional benchmark functions.

FNo	Name	D	Formula	C	Range	Fmin
F20	Box-Betts	3	$f(x)=\sum_{i=1}^{k}\left(e^{-0.1(i+1)x_1}-e^{-0.1(i+1)x_2}-\left[\left(e^{-0.1(i+1)}\right)-e^{-(i+1)x_3}\right]\right)^2$	NS, MM	[0.9, 1.2; 9, 11.2; 0.9, 1.2]	0
F21	Colville	4	$f(x)=100(x_1-x_2)^2+(x_1-1)^2+(x_3-1)^2+90\left(x_3^2-x_4\right)^2+10.1\left((x_2-1)^2+(x_4-1)^2\right)+19.8(x_2-1)(x_4-1)$	NS, MM	[−1,1]	0
F22	Csendes	4	$f(x)=\sum_{i=1}^{D}x_i^6(2+\sin\frac{1}{x_i})$	S, MM	[−1,1]	0
F23	Michalewicz	5	$f(x)=-\sum_{i=1}^{2}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	NS, MM	[0, π]	−4.6877
F24	Miele Cantrell	4	$f(x)=(e^{-x_1}-x_2)^4+100(x_2-x_3)^6+(\tan(x_3-x_4))^4+x_1^8$	NS, MM	[−1, 1]	0
F25	Step	5	$f(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	S, UM	[−100,100]	0
F26	Michalewicz	10	$f(x)=-\sum_{i=1}^{2}\sin(x_i)(\sin(ix_i^2/\pi))^{20}$	NS, MM	[0, π]	−9.6602
F27	Shubert	5	$f(x)=(\sum_{i=1}^{n}i\cos(i+1)x_i+i)(\sum_{i=1}^{n}i\cos(i+1)x_{i+1}+i)$	S, MM	[−10,10]	−186.73

energy generated from the individual sources may not be sufficient for a considerable amount of time. To efficiently utilize the renewable energy resources, a hybrid of these resources which minimizes the total annual cost (TAC) while providing optimum energy requirement will provide an acceptable solution to the demand for energy [73,74]. Here, this paper designed a hybrid system consisting of a PV generator, wind turbine generator and storage system. The schematic representation of the proposed hybrid system is given in Fig. 3. The DC bus helps to combine

Table 5
Thirty-dimensional benchmark functions.

FNo	Name	D	Formula	C	Range	Fmin
F28	Ackley	30	$f(x) = -20\exp[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{D}x_i^2}] - \exp[\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)] + 20 + e$	NS, MM	[−32,32]	0
F29	Brown	30	$\sum_{i=1}^{n-1}(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}$	NS, UM	[−1, 4]	0
F30	Ellipsoid	30	$f(x) = \sum_{i=1}^{n} i.x_i^2$	NS, UM	[−5.12,5.12]	0
F31	Griewank	30	$\frac{1}{4000} - 20\exp\left(\sum_{i=1}^{D}(x_i - 100)^2\right) - \left(\prod_{i=1}^{D}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right)\right) + 1$	NS, MM	[−100,100]	0
F32	Mishra	30	$f(x) = \left(1 + D - \sum_{i=1}^{N-1}x_i\right)^{N-\sum_{i=1}^{N-1}x_i}$	NS, MM	[0,1]	0
F33	Quartic	30	$f(x) = \sum_{i=1}^{D} ix_i^4 + random[0, 1)$	S, MM	[−1.28,1.28]	0
F34	Rastrigin	30	$f(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)]$	NS, MM	[−5.12,5.12]	0
F35	Rosenbrock	30	$f(x) = \sum_{i=1}^{D-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	NS, UM	[−30,30]	0
F36	Salomon	30	$f(x) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{30}x_i^2}) + 0.1\sqrt{\sum_{i=1}^{30}x_i^2}$	NS, MM	[−100,100]	0
F37	Sphere	30	$f(x) = \sum_{i=1}^{D} x_i^2$	S, MM	[−100,100]	0

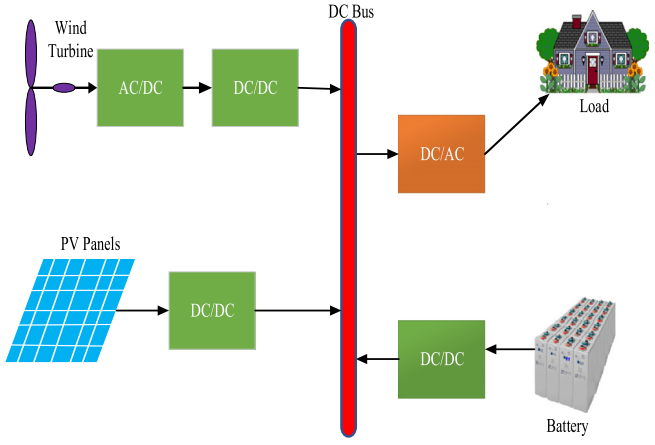

Fig. 3. Schematic of hybrid renewable energy resources.

the output power of the PV panels, wind turbines and batteries which supplied the load through a DC to AC converter.

The design is such that, if the power generated (P_{Gen}) by the hybrid resources at time t, is equal to the load demand (P_{Load}), the load demand is serviced by P_{Gen} through the inverters. If P_{Gen} is more than P_{Load}, the excess P_{Gen} will be stored in the battery bank. If the P_{Gen} is not enough to cater for P_{Load}, the battery bank which serves as storage will supply the shortage. If eventually P_{Load} is more than P_{Gen} and the storage system combine, the excess P_{Load} must be shed leading to power outage in the affected load. Detail information on the hybrid systems design are discussed in the following subsection.

4.2.1. Hybrid systems optimization problem formulation

To design an economic viable hybrid system, the objective function should minimize the total annual cost. The total annual cost (TAC) is the summation of annual capital cost (ACC) and annual maintenance cost (AMC). The TAC can be computed using Eq. (11)

$$TAC = \min\left(\sum_{j=1}^{N} ACC + \sum_{j=1}^{N} AMC\right) \tag{11}$$

where N is the number of optimization sample.

The capital cost which is also called design cost occurred at the beginning of the system design while the maintenance cost occurs during the project life span. The present value of the annuity which is a series of equal annual cash flows can be computed using the capital recovery factor (CRF) defined as a function of interest rate (a_r) by Eq. (12) [74,75]:

$$CRF = \frac{a_r(1 + a_r)^n}{(1 + a_r)^n + 1} \tag{12}$$

In this paper, the life span (n) of the hybrid system is selected as twenty-five years. Within this period, a battery maintenance schedule of every five years and inverter life span of fifteen years was set respectively. Therefore, the single payment battery present cost is computed by Eq. (13):

$$C_{Bat} = P_{Bat} \times \left(1 + \frac{1}{(1 + a_r)^5} + \frac{1}{(1 + a_r)^{10}} + \frac{1}{(1 + a_r)^{15}} + \frac{1}{(1 + a_r)^{20}} + \frac{1}{(1 + a_r)^{25}}\right) \tag{13}$$

Similarly, the single payment present worth of Converter/Inverter (C_{Inv}) is assumed to be the first 10 years and the last 15 years of the project life time.

$$C_{Inv} = P_{Inv} \times \left(1 + \frac{1}{(1 + a_r)^{10}} + \frac{1}{(1 + a_r)^{25}}\right) \tag{14}$$

where C_{Bat} is the present worth of battery and P_{Bat} is the battery price and C_{Inv} is the present worth of inverter/converter and P_{Inv} is the inverter/converter price.

The ACC is determined as the combined annual cost of the wind turbine, PV system, battery and inverter as in Eq. (15) [72,

76–78]:

$$ACC = CRF \times \left[n_{pv}C_{pv} + n_{wt}C_{wt} + n_{Bat}C_{Bat} + n_{Inv}C_{Inv}\right] \quad (15)$$

where n_{pv} is the number of PV panels, C_{pv} is the unit cost of PV panel, n_{wt} is the number of wind turbines, C_{wt} is the unit cost of wind turbine, n_{Bat} is the number of batteries, C_{Bat} is the present worth of battery, n_{Inv} is the number of converters/inverters and C_{Inv} is the present worth of converter/inverter.

The annual maintenance cost can be determined using Eq. (16)

$$AMC = n_{pv}C_{pv\_m} + n_{wt}C_{wt\_m} \quad (16)$$

where; C_{wt_m} and C_{pt_m} are the maintenance cost of wind turbine and PV panels respectively. The number of inverters n_{Inv} is selected as 4, as shown in the schematic diagram given in Fig. 5. When the sizing configuration is reduced to PV/Battery or Wind/Battery, the number of inverters/converters are reduced to 3.

4.2.2. Constraints definition

The optimization problem of the hybrid renewable system is to determine the right combination of PV panels, wind turbines and batteries which gives the minimum ACC and AMC. The ACC and AMC should, in turn, minimize the TAC given in Eq. (11) while satisfying the following constraints:

$$\begin{aligned} &n_{pv\_\min} \le n_{pv} \le n_{pv\_\max} \qquad n_{wt\_\min} \le n_{wt} \le n_{wt\_\max} \\ &n_{Bat\_\min} \le n_{Bat} \le n_{Bat\_\max} \\ &\text{and,} \\ &n_{pv},\ n_{wt} \text{ and } n_{Bat} = Integer \end{aligned} \quad (17)$$

From Eq. (21), $n_{pv\_\min}$ and $n_{pv\_\max}$ are the lower and upper limit of n_{pv}; $n_{wt\_\min}$ and $n_{wt\_\max}$ are the lower and upper bound of n_{wt} and, $n_{Bat\_\min}$ and $n_{Bat\_\max}$ are the lower and upper bound of n_{Bat}.

The charge quantity of the battery bank at any time should satisfy the following constraints

$$SOC(t_{\min}) \le SOC(t) \le SOC(t_{\max}) \quad (18)$$

The SOC whose maximum and minimum charge quantity is defined by $SOC(t_{\min})$ *and* $SOC(t_{\max})$ and is the battery State of Charge, determined by Eq. (19) [79–81]

$$SOC(t) = SOC(t-1)\times(1-\omega) + \left[\frac{P_L(t)}{\eta_{Inv}} - (P_{PV}(t) - P_{WT}(t))\right] \times \eta_{BC} \quad (19)$$

where; ω is the hourly self-discharge rate of the battery, η_{BC} is the battery bank discharge efficiency, η_{Inv} is the inverter efficiency.

The maximum charge quantity of the battery bank SOC (t_{max}) takes the value of nominal capacity of the battery (S_{Bat}) and the minimum charge quantity of the battery bank SOC (t_{min}) is obtained from the maximum Depth of Discharge (DOD) as in Eq. (20).

$$SOC(t_{\min}) = (1 - DOD) \times S_{Bat} \quad (20)$$

4.2.3. Photovoltaic system design

At any time, t, the output power of a PV panel P_{PV} can be determined from the solar radiation using Eq. (21)

$$P_{PV}(t) = I(t) \times A_{PV} \times \eta_{PV} \quad (21)$$

where I is the solar insolation (kW/m^2), A denotes the PV area (m^2) and η_{PV} is the efficiency of the PV's DC/DC converters. In this paper, we assumed that the PV panels is embedded with a maximum power point tracking (MPPT) algorithm.

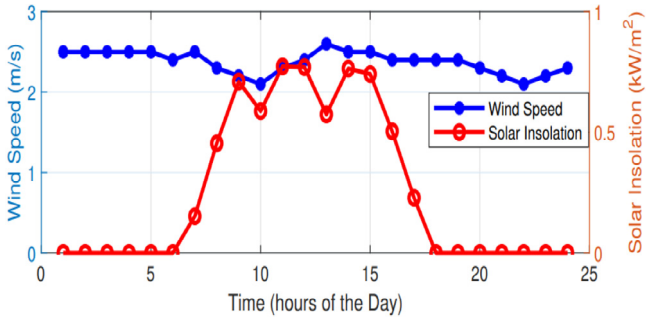

Fig. 4. Average hourly profiles of solar radiation and wind speed.

If the temperature effects on the PV panel output power are ignored, and the number of PV panels is n_{PV}, then, the overall output power by the PV system is given by Eq. (22):

$$P_{PV}(t) = n_{PV} \times P_{PV}(t) \quad (22)$$

4.2.4. Wind turbine

For a wind turbine, wind speed is the most critical factor. When the wind speed equals or exceeds the cut-in value, the wind turbine generator will start generating power. If the wind speed exceeds the cut-out value, the wind turbine generator stops running to protect the generator. The produced power of a wind turbine (P_{WT}) at time t is obtained as follows [82,83]:

$$P_{WT}(t) = \begin{cases} 0 & v_{co} \le v(t) \text{ or } v(t) \le v_{ci} \\ P_r \dfrac{v(t) - v_{ci}}{v_r - v_{ci}} & v_{ci} < v(t) < v_r \\ P_r & v_r < v(t) < v_{co} \end{cases} \quad (23)$$

where $v(t)$ is the wind speed at time t, v_{ci}, v_{co} and v_r are cut-in, cut-out and rated wind speed of the turbine respectively, Pr is the rated power of the wind turbine.

If the number of wind turbines is n_{WT}, the overall power produced by the wind turbine is:

$$P_{WT}(t) = n_{WT} \times P_{WT}(t) \quad (24)$$

4.2.5. Load profile

For a more realistic design, the essential parameters required for the hybrid system is taken from a practical study conducted for a case study area. Fig. 4, shows the model input data for solar radiation and wind speed, whereas, Fig. 5, shows the load demand for the designed case study area, a group of small households located at Abuja, Nigeria, with latitude: N 9°4′20.1504″ and longitude: E 7°29′28.6872″. The load profiles were modelled by administering questionnaires (see Appendix A) to take stock of installed electrical appliances. The questionnaires include information such as; kind of electrical appliance, power rating, make and total time of usage. This information is collated to computed the hourly load demand for the selected household for 24 h. To ensure a more realistic and stable design we employed the 24 h data which is synonymous to previous studies where data collected for a period of 24 h are mostly used.

4.2.6. System reliability and economic viability

Power systems reliability analysis plays a vital role in the hybrid energy system design process. This is because of the intermittent generation characteristics of the PV and WT. The hydride system reliability can be expressed in terms of Loss of Lower Supply Probability (LPSP). The LPSP is a metrics which is expressed as the ratio of the total hours of power failure over the full sample hours within a period [71]. When $P_{Gen}(t) < P_{Load}(t)$,

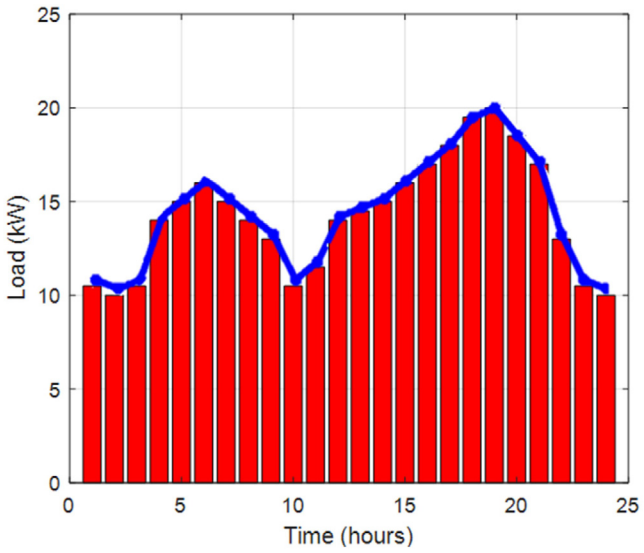

Fig. 5. Daily hourly load profile.

this places the hybrid energy system in a loss of Power Supply scenario. It should be noted that the lower the value of LPSP, the better the reliability of the hybrid system. The LPSP is expressed as [84]:

$$LPSP = \frac{\sum_{t=0}^{N} LPS(t)}{\sum_{t=0}^{N} P_{Load}(t)} \tag{25}$$

where; N = 24 i.e. the number of hours in a day, LPS is the loss of power supply given as:

$$LPS(t) = PLoad(t) - [P_{PV}(t) - P_{WT}(t) + P_{Bat}(t)] \times \eta_{Inv} \tag{26}$$

and $P_{Gen}(t)$ can be obtained by Eq. (27)

$$P_{Gen}(t) = P_{PV}(t) + P_{WT}(t) + P_{Bat}(t) \tag{27}$$

A LPSP of zero means that the load demand will always be satisfied and LPSP of one means that the load demand will never be satisfied. Therefore, the closer the value of LPSP to zero, the better the system.

To measure the average net present cost of the hybrid system, we compute the Levelized Cost of Energy (LCE) for generation system. The LCE is calculated as the ratio of the sum of cost over a life-time and the sum of energy produced over a lifetime [71]. The LCE is used to analyse the cheapest of all the hybrid system approaches implemented in the paper.

4.2.7. Energy generation flow analysis

In this section, we discuss the excess/deficit energy generated by the system. The designed system is an off-grid hybrid system where the renewable sources of energy are the only sources considered in catering for the load demand. Since the renewable sources are intermittent, an excess or a deficit of energy generation may occur. If this happens, the energy can be calculated by:

$$E_{e/d}(t) = P_{Gen}(t) - \frac{P_{Load}(t)}{n_{Inv}} \tag{28}$$

where $E_{e/d}(t)$ is the excess or deficit energy, generated at time t.

If $E_{e/d}(t) > 0$, it indicates there is an excess of renewable energy, however, if $E_{e/d}(t) < 0$ it shows there is a deficit of renewable energy. This implies that batteries start charging when there is an excess of energy.

5. Experimental results and discussion

This section provides a detailed comparison between the proposed algorithm and five (5) other algorithms namely; ABC, GA, PSO, CSA and OFA on thirty-seven (37) commonly used benchmark functions and the engineering optimization problem. The section is divided into two subsections. The first subsection discourses the solution accuracy and the convergence analysis of each algorithm on the benchmark functions. Whereas, the second subsection discourses the results obtained on the engineering optimization problem.

5.1. Results analysis on benchmark functions

This subsection is divided into two parts. The first part discusses the solution accuracy of each algorithms on the benchmark functions while the second part discusses the convergence analysis of the algorithms on the benchmark functions.

5.1.1. Solution accuracy study

According to the experiments carried out on the benchmark functions of Tables 3–5 the statistical results obtained for each algorithm over 30 independent runs are presented. Table 6, gives the results obtained for the 2D benchmark functions. The results obtained for 3D, 4D, 5D and 10D benchmark functions are presented in Tables 7 and 8, delineates the results obtained for 30D benchmark functions. For each benchmark functions, the best, average, standard deviation and rank of each algorithm are presented. In the tables, values in bold shows a comparatively best values, those in bold-italics represent a scenario where more than one algorithm obtained the best value. The performance of the algorithms is ranked using the best values of the benchmark functions. In a situation where more than one algorithm brings the same best value, we break the tie using the mean value or standard deviation. If *n* number of algorithms obtain the same best value in any of the benchmark functions, these algorithms are ranked the same and the next best performing algorithm is ranked n+1. At the end of each table, we compute the average rank of each algorithm on the benchmark function. This average rank was used to decide the final rank of the algorithms.

From the results of the 2-Dimensional function listed in Table 6, it can be observed that, the SAO algorithm obtained the global solution in 15 benchmark functions (F1, F3, F4, F6, F7, F8, F11, F12, F13, F14, F15, F16, F17, F18 and F19). The ABC, CSA, GA PSO, OFA and AOA also obtained the global solution for 8 (F6, F10–F13, F16, F17, F19), 4 (F6, F11, F13, F19), 8 (F1, F4, F6, F10, F11, F13, F17, F17), 8 (F1, F4, F10, F11, F13, F17, F19), 2 (F4, F11) and 7 (F1, F2, F8, F14, F17, F18 and F19) benchmark functions respectively.

The results in Table 6, also showed that, none of the algorithms obtained the global solution for F2, F5 and F9 benchmark functions. For these functions, the PSO got the best results for F2 and F5, whereas, the SAO attained the best result for F9. The rank of the algorithm showed the SAO obtained final rank value of 1, indicating its superior performance over the other algorithms on the 2D benchmark functions. The ranking obtained for ABC, CSA, GA, PSO, OFA and AOA are 4, 6, 3, 2, 7 and 5 respectively.

From the statistical results of 3D, 4D, 5D and 10D functions given in Table 7, it is observed that SAO obtained the global solutions in 6 (F20, F21, F23–F27) of these functions group, this is more than the global solution obtained by any of the other algorithm. Similarly, the ABC, CSA, GA, PSO, OFA and AOA obtained the global solutions on 1 (F25), 1 (F25), 2 (F25, F27) 3 (F21, F25, F27), 1 (F25) and 2 (F22 and F23) respectively. For these cluster of benchmark functions, none of the algorithms obtain the global solution for F22 and F24 functions. However, the PSO obtained

Table 6
Results comparison on 19 2D benchmark functions.

Fn		ABC	CSA	GA	PSO	OFA	SAO	AOA
F1	Avg	−6.2071	−3.538	−2.0218	−2.0218	−1.5976	−2.0203	−1.5393
	Best	−4.7422	−4.043	***−2.0218***	***−2.0218***	−1.824	**−2.0218**	**−2.0218**
	Std	11.04	3.15E−00	4.44e−16	0.00	2.23E−01	2.48E−03	4.83E−01
	Rank	7	6	2	1	5	3	4
F2	Avg	2.38E−06	1.00E−02	1.30E−02	1.52E−07	1.15E−04	3.19E−04	6.78E−01
	Best	3.22E−08	4.60E−03	1.29E−04	**1.51E−21**	6.57E−07	5.87E−05	1.84E−04
	Std	2.93E−06	5.15E−03	1.40E−02	3.05E−07	1.15E−04	3.54E−04	1.23E+00
	Rank	2	7	5	1	3	4	6
F3	Avg	−106.787	−106.433	−106.790	−106.790	−101.635	−106.768	−105.333
	Best	−106.763	−106.761	−106.762	−106.763	−105.826	**−106.765**	−106.763
	Std	5.51E−05	3.36E−01	1.42E−14	1.42E−14	4.19	8.77E−16	4.36E+00
	Rank	2	5	4	3	7	1	6
F4	Avg	1.61E−09	1.21E−02	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
	Best	1.37E−13	4.14E−05	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***
	Std	2.57E−09	2.03E−02	0.00	0.00	0.00	0.00	0.00
	Rank	6	7	1	1	1	1	1
F5	Avg	1.99E−08	1.77E−02	4.36E−03	1.26E−21	3.10E+00	4.73E−03	5.05E−03
	Best	8.19E−09	3.12E−05	1.22E−05	**1.10E−23**	8.53E−02	8.82E−04	1.34E−03
	Std	7.89E−09	1.43E−02	9.55E−03	2.91E−21	3.43E+00	2.93E−03	3.67E−03
	Rank	2	5	3	1	7	4	6
F6	Avg	*0.3979*	0.3999	*0.3979*	1.7952	0.4599	*0.3979*	0.5131
	Best	***0.3979***	***0.3979***	***0.3979***	0.4636	0.4021	***0.3979***	0.3982
	Std	3.86E−06	1.44E−03	1.72E−07	0.444	8.78E−02	4.20E−08	3.02E−01
	Rank	3	4	2	7	6	1	5
F7	Avg	−0.2079	−0.0046	−0.2382	−0.1982	−0.0178	−0.1779	−0.2016
	Best	−0.2773	−0.0091	−2.7775	−0.2778	−0.0177	**−0.1799**	−0.2753
	Std	5.09E−02	1.37E−03	5.86e−02	7.00e−02	3.35E−05	1.91E−03	7.04E−02
	Rank	4	7	5	2	6	1	3
F8	Avg	1.31E−23	5.95E−06	4.80E−52	*1.38E−87*	2.06E−01	*1.38E−87*	3.17E−45
	Best	1.72E−25	1.10E−08	7.07E−60	***1.38e−87***	9.70E−04	***1.38E−87***	**2.91E−80**
	Std	2.26E−23	1.15E−05	8.22E−52	0.00	2.64E−01	3.11E−86	9.51E−45
	Rank	5	6	4	1	7	1	3
F9	Avg	4.71E−01	6.25E−01	9.90E−02	9.87E−02	1.39E−01	2.49E−04	1.43E+00
	Best	1.26E−01	1.12E−01	1.02E−01	3.66E−02	1.80E−01	**1.44E−06**	1.00E−01
	Std	2.37E−01	3.31E−01	4.59E−03	3.72E−07	3.87E−02	1.70E−04	3.99E+00
	Rank	4	5	3	2	6	1	7
F10	Avg	−1.0316	−1.0310	−1.0316	−1.0316	−0.6117	−1.0316	−0.9556
	Best	***−1.0316***	−1.0315	***−1.0316***	***−1.0316***	−0.9740	***−1.0316***	−1.0185
	Std	6.66E−08	5.13E−04	2.22E−16	0.00	2.57E−01	0.00	4.18E−02
	Rank	4	5	3	1	7	1	6
F11	Avg	−42.935	−42.901	−42.855	−42.944	−38.589	−42.948	−27.628
	Best	***−42.944***	***−42.944***	***−42.944***	**−42.944**	***−42.944***	***−42.944***	−28.472
	Std	4.01E−03	2.72E−02	1.79E−01	0.00	4.22	1.41E−05	2.29E+00
	Rank	3	5	4	1	6	2	7
F12	Avg	−24776	−24063	−224737	−24766	−15350	−24776	−24736
	Best	***−24777***	−24762	−24776	−24766	−24434	***−24777***	−24776
	Std	0.64	981.73	612	0.00	6.56E+03	1.70E−18	1.20E+02
	Rank	2	5	6	4	7	1	3
F13	Avg	−1.00	−0.84	−1.00	−1.00	−0.21	−1.00	−0.10
	Best	**−1.00**	**−1.00**	***−1.00***	***−1.00***	−0.79	***−1.00***	−0.99
	Std	3.90E−02	1.40E−01	0.00	0.00	2.63e−01	0.00	3.00E−01
	Rank	4	5	1	1	7	1	6
F14	Avg	6.88E−08	6.75E−05	3.63E−10	3.72E−19	3.34E−07	0.00	0.00
	Best	5.38E−09	6.76E−06	1.51E−15	2.37E−21	3.23E−09	**0.00**	**0.00**
	Std	8.79E−08	5.46E−05	8.11E−10	3.82E−19	5.64E−07	0.00	0.00
	Rank	5	7	4	3	6	1	1
F15	Avg	−19.671	−69.1009	−33.681	−37.022	−33.4397	−1.9131	−36.3228
	Best	−14.008	−86.0503	−31.914	−39.267	−37.4028	**−1.9132**	−39.0000
	Std	7.281	8.074	2.75e−01	2.34	2.0818	5.64E−05	3.11E+00
	Rank	2	7	3	6	4	1	5
F16	Avg	−1.9999	−1.9999	−2.00	−2.00	−1.9107	−1.8028	−1.9990
	Best	***−1.8013***	−1.9999	−2.00	−2.00	−1.9996	***−1.8013***	−1.9999
	Std	6.14E−08	4.29E−05	0.00	0.00	1.41E−01	2.05E−03	1.98E−03
	Rank	2	5	6	6	3	1	4
F17	Avg	−3873.72	−3873.60	−3873.72	−3873.72	−3639.75	−3873.72	−3873.72
	Best	***−3873.72***	−3873.71	***−3873.72***	***−3873.72***	−3850.41	***−3873.72***	***−3873.72***
	Std	0.00	1.13e−01	0.00	0.00	151	0.00	1.82E−05
	Rank	1	6	1	1	7	1	1
F18	Avg	1.04E−06	3.78E−07	2.56E−17	1.46E−08	2.24E−04	0.00	0.00
	Best	1.59E−09	7.17E−14	1.79E−28	5.24E−24	2.33E−07	**0.00**	**0.00**
	Std	1.29E−06	6.63E−07	7.23E−07	4.38E−09	1.69E−04	0.00	0.00
	Rank	6	5	4	3	7	1	1

(*continued on next page*)

Table 6 (*continued*).

Fn		ABC	CSA	GA	PSO	OFA	SAO	AOA
F19	Avg	−78.332	−78.280	−78.332	−78.332	−74.528	−78.332	−75.663
	Best	***−78.332***	**−78.332**	***−78.332***	***−78.332***	−77.950	***−78.332***	***−78.332***
	Std	0.00	4.25E−02	0.00	0.00	4.00+00	0.00	4.19E+00
	Rank	1	5	1	1	7	1	6
Count of global best		**8**	**4**	**8**	**8**	**2**	**15**	**7**
Average rank		**3.42**	**5.63**	**3.26**	**2.42**	**5.74**	**1.47**	**4.23**
Final rank		**4**	**6**	**3**	**2**	**7**	**1**	**5**

Table 7
Results comparison on 3D, 4D, 5D and 10D Benchmark functions.

Fn		ABC	CSA	GA	PSO	OFA	SAO	AOA
F20	Avg	9.26E−08	7.58E−06	1.26E+02	5.45E−13	6.32E−01	0.00	1.01E+02
	Best	5.40E−08	2.76E−07	1.26E+02	3.37E−20	6.00E−01	**0.00**	1.05E−01
	Std	7.28E−07	5.487E−06	4.78E−07	1.63E−12	1.91E−02	0.00	5.01E+1
	Rank	3	4	7	2	5	1	6
F21	Avg	9.86E−01	3.85E−01	4.02	0.00	7.20E−01	0.00	1.09E+01
	Best	6.02E−02	2.19E−01	1.03E−01	***0.00***	3.15E−01	***0.00***	1.57E+01
	Std	8.57E−01	1.71E−01	3.43	0.00	3.91E−01	0.00	3.21E+00
	Rank	3	5	4	1	6	1	7
F22	Avg	2.14E−15	1.41E−04	9.99E−16	9.50E−56	2.73E−05	5.34E−30	0.00
	Best	1.30E−15	1.96E−07	5.86E−24	2.91E−62	8.19E−09	1.27E−33	**0.00**
	Std	3.63E−07	3.11E−04	2.90E−15	2.82E−55	5.83E−05	1.29E−29	0.00
	Rank	5	7	4	2	6	3	1
F23	Avg	−4.9997	−5.00	−5.00	−5.00	−4.5224	−4.6864	−4.9915
	Best	−4.6877	−5.00	−5.00	−5.00	−4.9647	**−4.6877**	−5.000
	Std	2.57E−04	1.59E−03	0.00	0.00	3.66E−01	2.21E−03	2.07E−02
	Rank	2	6	5	5	3	1	4
F24	Avg	4.64E−06	5.31E−05	2.48E−04	7.00E−13	4.00E−03	2.40E−16	7.21E−04
	Best	2.65E−07	3.89E−06	2.11E−06	2.37–18	2.56E−03	**2.06E−19**	1.84E−04
	Std	6.19E−06	4.80E−05	2.34E−04	1.18E−12	2.49E−03	3.08E−11	4.26E−04
	Rank	3	4	5	2	7	1	6
F25	Avg	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Best	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Rank	1	1	1	1	1	1	1
F26	Avg	−9.9999	−9.9980	−10.00	−10.00	−9.5391	−9.6618	−9.9825
	Best	−9.9991	−9.9942	−10.00	−10.00	−9.6682	**−9.6601**	−9.8720
	Std	2.80E−04	1.68E−03	0.00	0.00	3.56E−01	2.63E−03	3.89E−02
	Rank	5	4	6	7	2	1	3
F27	Avg	−186.37	−186.59	−186.73	−186.73	−349.88	−186.73	−110.51
	Best	−186.71	−186.71	***−186.73***	***−186.73***	−359.25	***−186.73***	−166.61
	Std	4.33E−01	1.01E−01	5.75E−04	2.84E−14	9.87	8.01E−12	3.19E+01
	Rank	5	4	3	1	6	2	7
Count of global best		**1**	**1**	**2**	**3**	**1**	**6**	**2**
Average rank		**3.38**	**4.38**	**4.38**	**2.63**	**4.50**	**1.38**	**4.38**
Final rank		**3**	**4**	**4**	**2**	**7**	**1**	**4**

the best results for F22 whereas the SAO obtained the best results for F24. The ranking of the algorithms also showed that the SAO has the best performance for these group of functions by obtaining a final rank value of 1. The PSO ranked second with a rank value of 2 while the ABC ranked third with a rank value of 3. However, the CSA, GA and AOA obtained the same rank with a final rank value of 4. Whereas the OFA obtained the least performance for these group functions by obtaining a rank value of 7. Note that, because CSA, GA and AOA all obtained the same (4), the ranking 5 and 6 was not used accordingly.

Table 8 shows the statistical results obtained for 30-dimensional benchmark functions. From this table, it can be observed that, the SAO obtain the global results in 7 (F29–F32, F34, F36–F37) out of the ten 30D benchmark functions. For this group of functions, results also showed that, ABC, GA, PSO, OFA and AOA obtained the global solutions in 3 (F30, F31, F32), 2 (F31, F34), 3 (F31, F32, F34), 3 (F29, F32, F36) and 6 (F29–F31, F34, F36–F37) benchmark functions respectively. However, the CSA algorithm could not obtain the global solution to any of the functions in these group. For F28, F33 and F35 where none of the algorithm obtain the global solution, it was observed that the PSO got the overall best solution for F35, whereas, OFA, SAO and AOA obtained the same best result for F28 and only AOA obtain the best result for F33 function. The algorithms ranking indicate that SAO has the best ranking by obtaining a final rank value of 1. The final ranking for the other algorithms' ABC, CSA, GA, PSO, OFA and AOA were computed as 3, 7, 5, 4, 6 and 2 respectively.

5.1.2. Non-parametric study and robustness

Since the process of metaheuristics algorithms starts by random initialization of potential solutions, their performance is best described through multiple simulations. For this reason, we analyse the general performance of SAO through 30 independent runs to compute the average, best and standard deviation of each benchmark functions. To measure the robustness and verify whether the results of SAO are significantly better than the results of the other algorithms, we performed Wilcoxon rank-sum non-parametric test. The results of this test given in Table 9, was performed for benchmark functions where none of the algorithms obtains the global result.

As stated earlier, the Wilcoxon test was implemented at a 5% ($p = 0.05$) significant probability level. When the p-value $<$ 0.05 and the null hypothesis, h-value $= 1$, then, the difference in solution sets obtained by two algorithms is significant. However,

Table 8
Results comparison on 30D benchmark functions.

Fn		ABC	CSA	GA	PSO	OFA	SAO	AOA
28	Avg	2.15E−14	1.53E−02	5.30E−13	6.28E−12	7.88E−03	8.88E−16	9.11E−16
	Best	7.99E−15	1.42E−03	4.48E−12	2.29E−12	***8.88E−16***	***8.88E−16***	**8.88E−16**
	Std	1.00E−14	1.66E−02	5.75E−11	4.21E−12	1.30E−02	0.00	4.47E−17
	Rank	4	7	5	6	3	1	2
F29	Avg	1.47E−29	1.83E−03	4.96E−08	5.99E−25	*0.00*	*0.00*	*0.00*
	Best	1.22E−31	3.94E−05	2.03E−25	1.77E−27	***0.00***	***0.00***	***0.00***
	Std	3.29E−29	1.35E−03	1.24E−07	1.61E−24	0.00	0.00	0.00
	Rank	4	7	6	5	1	1	1
F30	Avg	*0.00*	3.16E−07	1.11E−62	9.37E−28	1.60E−10	*0.00*	*0.00*
	Best	***0.00***	4.50E−10	2.10E−97	1.81E−31	1.52E−13	***0.00***	***0.00***
	Std	0.00	4.32E−07	3.34E−62	1.47E−27	2.00E−10	0.00	0.00
	Rank	1	7	4	5	6	1	1
F31	Avg	0.00	5.37E−03	0.00	0.00	2.02E−03	0.00	0.00
	Best	***0.00***	4.95E−03	***0.00***	***0.00***	1.36E−02	***0.00***	***0.00***
	Std	0.00	2.42E−04	0.00	0.00	2.45E−01	0.00	0.00
	Rank	1	6	1	1	7	1	1
F32	Avg	0.00	7.03E−05	30.1	0.00	0.00	0.00	5.21E+00
	Best	***0.00***	1.11E−06	14.30	***0.00***	***0.00***	***0.00***	2.21E+00
	Std	0.00	1.23E−04	11.40	0.00	0.00	0.00	2.09E+00
	Rank	1	5	7	1	1	1	6
F33	Avg	1.7E+01	5.83E−01	1.91E+01	5.55E−04	1.13E+02	2.77E−01	2.80E−04
	Best	1.08E+01	4.10E−01	4.64E+01	1.03E−04	6.30E+01	3.58E−02	**1.49E−05**
	Std	4.44E+00	1.45E−01	2.13E+02	6.26E−04	2.13E+02	2.74E−01	2.51E−04
	Rank	5	4	6	2	7	3	1
F34	Avg	9.54E−14	3.78E−02	0.00	0.00	5.86E+00	0.00	0.00
	Best	3.55E−15	1.22E−03	***0.00***	***0.00***	1.12E+00	***0.00***	***0.00***
	Std	2.10E−13	2.49E−02	0.00	0.00	3.49E+00	0.00	0.00
	Rank	5	6	1	1	7	1	1
F35	Avg	2.00E−03	5.48E−02	6.95E−01	**4.94E−07**	1.37	8.19E−03	6.84E−02
	Best	1.96E−04	4.75E−02	2.93E−03	**7.13E−10**	3.59E−01	6.91E−02	8.65E−03
	Std	1.84E−03	5.50E−03	1.03	1.23E−06	5.87E−01	6.76E−02	2.16E−02
	Rank	2	6	5	1	7	4	3
F36	Avg	2.88E−21	1.05E−04	4.93E−47	1.41E−08	0.00	0.00	0.00
	Best	2.88E−21	1.06E−07	9.47E−52	1.68E−12	***0.00***	***0.00***	***0.00***
	Std	0.00	2.79E−04	1.25E−46	4.24E−12	0.00	0.00	0.00
	Rank	4	6	3	5	1	1	1
F37	Avg	6.69E−55	1.12E−08	6.33E−08	5.36E−28	7.32E−06	0.00	0.00
	Best	6.85E−59	4.27E−11	2.15E−09	4.24E−32	2.74E−09	**0.00**	**0.00**
	Std	1.08E−54	1.88E−08	7.62E−08	1.39E−27	1.01E−05	0.00	0.00
	Rank	3	5	6	4	7	1	1
Count of global best		**3**	**0**	**2**	**3**	**3**	**7**	**6**
Average rank		**3.0**	**5.3**	**4.4**	**3.1**	**4.7**	**1.5**	**1.8**
Final rank		**3**	**7**	**5**	**4**	**6**	**1**	**2**

Table 9
Wilcoxon's rank sum test.

Functions		SAO Vs.					
Fn	Wilcoxon	ABC	CSA	GA	PSO	OFA	AOA
F2	p-value	1.8267e−04	1.8267e−04	0.0046	1.8165e−04	0.0890	0.0013
	h-value	1	1	1	1	0	1
F5	p-value	1.8267e−04	0.0235	0.0169	1.6170e−05	6.2521e−05	0.9097
	h-value	1	1	1	1	1	0
F8	p-value	2.3960e−08	6.2602e−12	6.2602e−12	0.1681	2.3960e−08	1.1067e−04
	h-value	1	1	1	0	1	1
F9	p-value	6.6063e−08	6.1935e−11	2.3783e−11	2.8718e−11	2.8234e−11	8.7450e−05
	h-value	1	1	1	1	1	1
F22	p-value	1.9647e−11	1.8267e−04	1.2353e−04	6.6063e−08	4.5088e−08	6.3864e−05
	h-value	1	1	1	1	1	1
F24	p-value	6.6063e−08	1.8267e−04	1.8267e−04	0.0058	1.8267e−04	1.8267e−04
	h-value	1	1	1	1	1	1
F28	p-value	6.2487e−05	6.3864e−05	6.3864e−05	0.0149	6.3403e−05	0.1675
	h-value	1	1	1	1	1	0
F33	p-value	1.8267e−04	0.0140	1.8165e−04	1.8267e−04	1.8267e−04	1.8267e−04
	h-value	1	1	1	1	1	1
F35	p-value	1.8267e−04	0.5521	0.9698	1.1744e−05	1.1744e−05	0.0312
	h-value	1	1	0	1	1	1

if the p-value > 0.05 and the h-value $= 0$, then, the difference in the solution sets obtained by the two algorithm is not significant. The rank-sum test indicates that the results of SAO for functions where non of the algorithms obtained the global solution dominates the results of the other algorithms. Similarly, for functions where any of the other algorithms obtained the best results, the rank-sum test also indicates statistically significant results for such an algorithm. In all the functions considered for this test, the h-value obtained shows a value of 1, except for F2, F5, F8, F28 and F35 where OFA, PSO, GA and AOA obtained h-values 0.

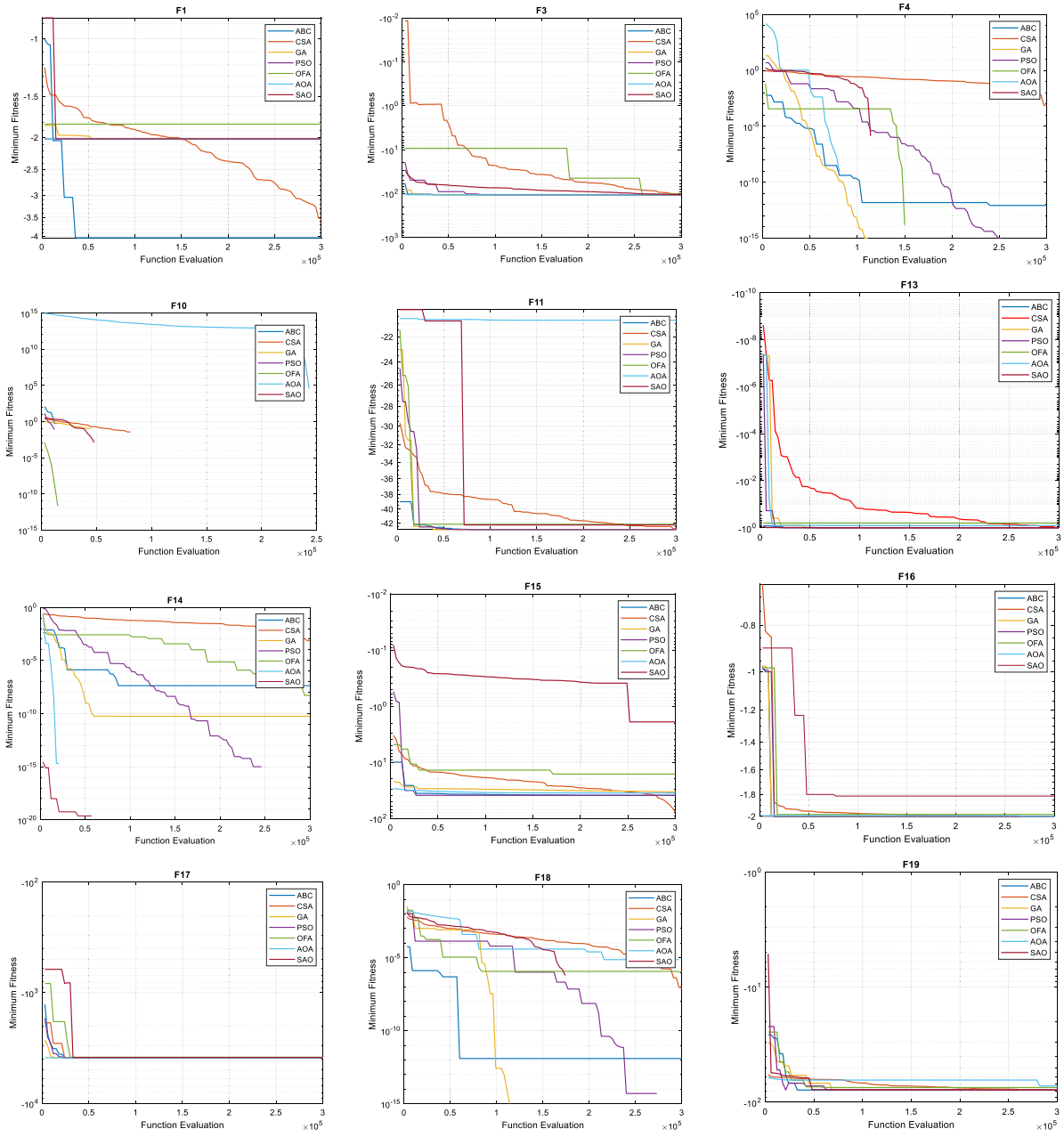

Fig. 6. Two-dimensional benchmark functions.

5.1.3. Convergence study

Due to the specific characteristics of every metaheuristic algorithm, its often not advisable to evaluate their convergence and execution speed using only iteration. Iteration will create an uneven basis for comparing the performance of the algorithms because some algorithm could evaluate the objective function more than twice in every iteration whereas, others may evaluate the objective function only ones. For example, the new algorithm (SAO) presented in this paper evaluate the objective function four times in every iteration, whereas the reference algorithms; CSA-GA-PSO, OFA, ABC and AOA required the evaluation of objective function, two, three, four and five times in every iteration respectively. To ensure equal bases for comparison, number of function evaluation prove to be a better alternative. Using function evaluation, the termination criteria for each algorithm can be set to equal number of function evaluation (say 300,000 as in the case of this paper). This means, the optimization process of each algorithm will terminate when the objective function has been evaluated 300,000 times. This way, all the algorithm will have equal opportunity to search for the global optimum irrespective of number of iterations.

For uniform and fair analysis of the convergence of each algorithm, we first superimposed the convergence plots on some randomly selected (24) benchmark functions from the group of functions given in this paper. The selection was partly influence by how close the solution is to the global optimum. Since the benchmark functions are classified into three categories, we present the figures according to these classifications. For example, Fig. 6, present the convergence plots of SAO and the other algorithms for 2-D benchmark problems. From these figures, it can be observed that the SAO converges faster in a fewer number of FE than most of the other algorithms. This assertion is justified by the FE and AR values given in Table 10 The table showed that, the AR $>$ 1 in all the algorithms for F10, F13, F14, F15 and F18.

Table 10
Selected 2D benchmark functions.

Fn	SAO	ABC	AR	CSA	AR	GA	AR	PSO	AR	OFA	AR	AOA	AR
F1	20 000	300 000	15.00	300 000	15.00	55 000	2.75	5000	0.25	300 000	15.00	42 000	2.10
F3	300 000	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	53 000	0.17
F4	118 000	300 000	2.54	300 000	2.54	110 000	0.93	250 000	2.12	290 000	2.46	78 000	0.66
F10	48 000	110 000	2.29	300 000	2.29	410 000	8.54	120 000	2.50	220 000	2.18	300 000	6.25
F11	70 000	18 000	0.26	290 000	4.14	50 000	0.71	60 000	0.86	30 000	0.43	300 000	4.29
F13	5000	20 000	4.00	290 000	58.00	48 000	9.60	20 000	4.00	300 000	60.00	20 000	4.00
F14	40 000	300 000	7.50	300 000	7.50	300 000	7.50	300 000	7.50	300 000	7.50	21 000	0.52
F15	251 000	300 000	1.20	300 000	1.20	300 000	1.20	300 000	1.20	300 000	1.20	150 000	0.60
F16	70 000	10 000	0.14	300 000	4.29	300 000	4.29	300 000	4.29	300 000	4.29	30 000	0.43
F17	30 000	25 000	0.83	300 000	10	10 000	0.33	25 000	0.83	300 000	10.00	20 000	0.67
F18	170 000	300 000	1.76	300 000	1.76	300 000	1.76	300 000	1.76	300 000	1.76	222 000	1.31
F19	49 000	30 000	0.61	250 000	5.10	70 000	1.43	60 000	1.22	300 000	6.12	280 000	5.71
	Avg AR	**3.10**		**9.73**		**3.34**		**2.30**		**9.33**		**2.22**	**3.10**

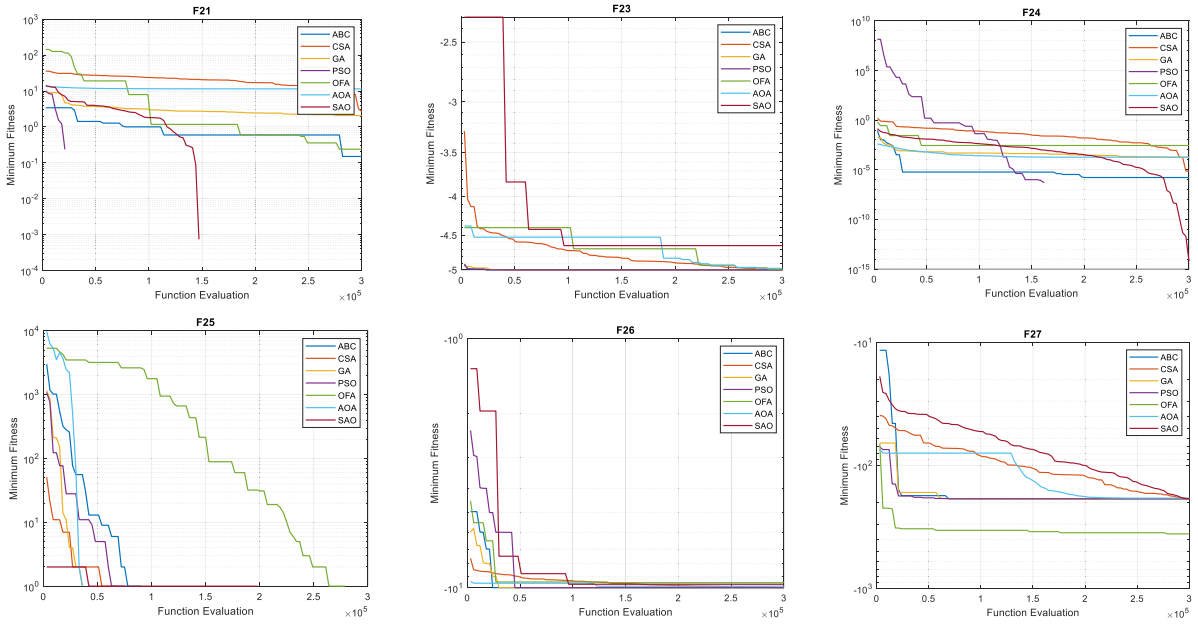

Fig. 7. Three, four, five and Ten-dimensional benchmark functions.

Table 11
Selected three, four, five and ten-dimensional benchmark functions.

Fn	SAO	ABC	AR	CSA	AR	GA	AR	PSO	AR	OFA	AR	AOA	AR
F21	290 000	300 000	1.03	300 000	1.03	300 000	1.03	60 000	0.21	300 000	1.03	300 000	1.03
F23	95 000	300 000	3.16	300 000	3.16	300 000	3.16	300 000	3.16	300 000	3.16	300 000	3.16
F24	300 000	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00
F25	40 000	70 000	1.75	53 000	1.32	30 000	0.75	60 000	1.50	260 000	6.50	41 000	1.03
F26	95 000	300 000	3.16	300 000	3.16	300 000	3.16	300 000	3.16	300 000	3.16	300 000	3.16
F27	290 000	300 000	1.03	300 000	1.03	65 000	0.22	60 000	0.21	300 000	1.03	300 000	1.03
	Avg AR		**1.86**		**1.79**		**1.55**		**1.54**		**2.65**		**1.74**

This indicates the superior convergence of SAO over the other algorithms on these benchmark functions. The table also showed that in few numbers of cases, an AR $<$ 1 was obtained by some algorithms. When compare with AOA, the AR $<$ 1 was obtained for F3–F4 and F14–F18. This implies that, for these functions the AOA has a faster convergence.

The convergence plots for some selected 3D, 4D, 5D and 10D functions are given Fig. 7 and the acceleration rate analysis are shown in Table 10. From Table 11, it can be observed that, SAO converged to its optimum solution in fewer FE for F23 and F26 by indicating an AR $>$ 1 for all the algorithms. In F21, the AR values showed that, PSO has a faster convergence, whereas, the GA converges faster in F25 by respectively indicating an AR $<$ 1. However, both GA and PSO convergences to its optimum results in fewer FE in F27 compared to the SAO. All the algorithm has the same convergence in F24, leading to an AR $=$ 1. Comparing the convergence of SAO with that of AOA for these group of functions, the AR $>$ 1 was obtained.

Similarly, in the case of 30-dimensional benchmark functions, the convergence plot given in Fig. 8 shows that the SAO also converges to its optimum solution in a fewer number of FE in all the benchmark functions except in F36 and F37 where OFA and GA has a faster convergence respectively. However, all the algorithms have equal convergence in F28 and F35. This convergence is supported by the AR values given in Table 12. From this table, it can be seen that FE values of SAO are better than that of the other algorithms except in a few cases highlighted just above. When compared to AOA, the SAO only converges faster in F35 and F37, the AOA converges faster in the other algorithms.

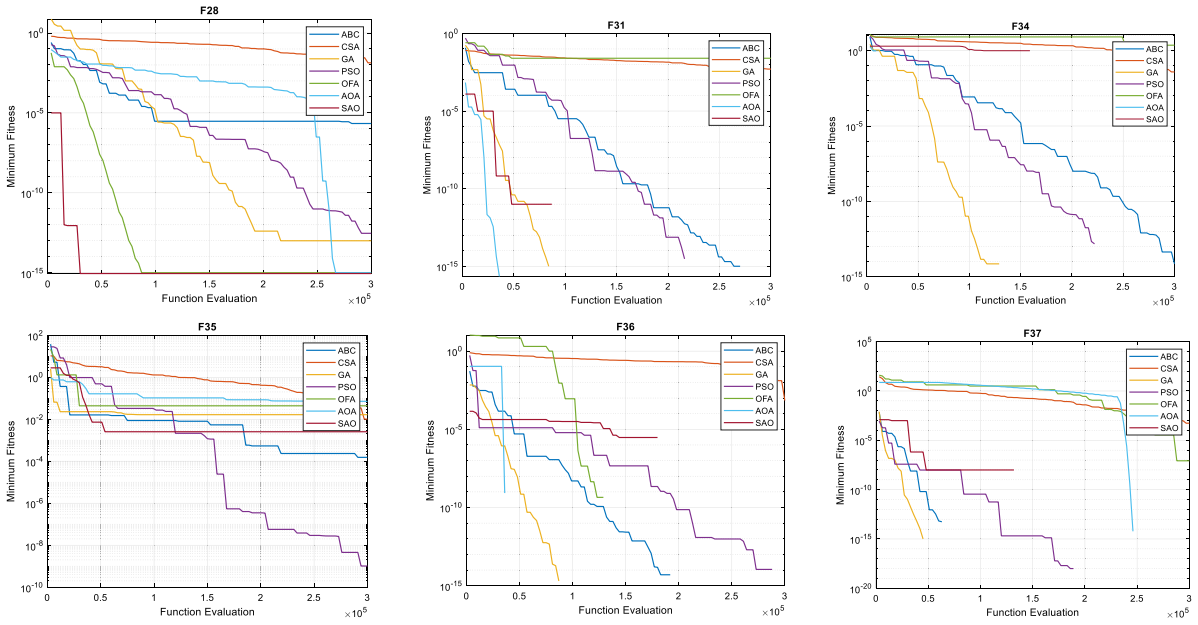

Fig. 8. Thirty-dimensional benchmark functions.

Table 12
Selected thirty dimensional functions.

Fn	SAO	ABC	AR	CSA	AR	GA	AR	PSO	AR	OFA	AR	AOA	AR
F28	300 000	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	265 000	0.88
F31	50 000	260 000	5.20	300 000	6.00	80 000	1.60	220 000	4.40	300 000	6.00	40 000	0.80
F34	110 000	300 000	2.73	300 000	2.73	120 000	1.09	210 000	1.91	300 000	2.73	30 000	0.27
F35	300 000	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00	300 000	1.00
F36	50 000	300 000	6.00	300 000	6.00	300 000	6.00	300 000	6.00	30 000	0.60	40 000	0.88
F37	50 000	300 000	6.00	300 000	6.00	30 000	0.60	300 000	6.00	300 000	6.00	249 800	4.99
	Avg AR		**3.65**		**3.79**		**1.88**		**3.38**		**2.89**		**1.47**

5.2. Results analysis of hybrid renewable energy systems

This section presents and analyses the results of the sizing of hybrid renewable energy systems. The parameters of each algorithm presented in Table 2, were used in the design of the hybrid system. The lower and upper bound of each decision variables (n_{pv}, n_{wt} and n_{bat}) were set to 1 and 250 respectively. Tables 13 and 14 give a summary of the results obtained by the algorithms. Table 13 shows the results of the hybrid system for three different combinations of the renewable resources. The first combination considers PV, Wind and Battery system (PV/Wind/Battery), the second combination consider PV and Battery system (PV/Wind). In contrast, the third combination considers Wind and Battery (Wind/Battery) system. The best results obtained by each sizing combinations are presented in Table 13, while the average (Avg), best (Best) and standard deviation (Std) performed over 30 independent runs are summarized in Table 14.

From Table 13, it can be observed that the PV/Battery is the most economical system to supply the load demand by SAO based methods. The Total Annual Cost (TAC) of PV/battery system obtained by SAO is 8016.96$, while the TAC for Wind/Battery, and PV/Wind/Battery systems were 8771.18$ and 10,526.44$ respectively. The table showed that SAO produce more economical results than the other algorithms for PV/Wind/Battery system except for OFA which was marginally better than SAO. The optimize sizing obtained by SAO for each sizing combination is, PV/Wind/Battery: $N_{PV} = 210$, $N_{WT} = 44$, $N_{Batt} = 44$, $N_{Conv/Inv} = 4$; PV/Battery: $N_{PV} = 187$, $N_{WT} = 0$; $N_{Batt} = 25$, $N_{Conv/Inv} = 3$; Wind/Battery: $N_{PV} = 0$, $N_{WT} = 25$; $N_{Batt} = 105$, $N_{Conv/Inv} = 3$. The cost of PV, Wind, Battery and Converter/Inverter system are also given in Table 13.

To correctly measure the stability of each algorithm on the design of the hybrid systems, we calculate the rank of there on each combination of the hybrid. This rank, which can be seen in Table 14, was calculated using the best TAC obtained by each algorithm on the hybrid system. From these ranks, it can be observed that both SAO and GA were ranked 2nd, only to OFA. However, the final ranks obtained by ABC, CSA, PSO and AOA are 5, 7, 4 and 6 respectively.

The breakdown of the total annual cost obtained by each algorithm is shown in Figs. 9–14. To analyse the economic viability and possibility of adopting our proposed hybrid systems, we compute the LSPS and the Levelized Cost of Energy (LCE) for each combination of the hybrid system. The LPSP helps to determine the point, at which the energy generated, will not be enough to cater to the load demand of customers. The closer the LPSP to zero, the better the hybrid system. The LCE helps to measure the average revenue per unit of energy generated, which will help to recover the cost of the hybrid system over its life time. For example, the LCE can be thought of as the average minimum cost for which the generated energy by the hybrid systems can be sold for, to offset the total installation cost over its life cycle. Table 14, shows the average LSPS, LCE and $E_{e/d}$ obtained by each algorithm on the hybrid systems design combinations. From Table 15, it can be seen that the installation cost for PV/Wind/Battery, PV/Battery and Wind/Battery can be recovered within the project life span by selling the energy for 1.45 $/kW, 1.20 $/kW and 1.23 $/kW respectively using the SAO sizing methods. These is by far the most economic design considering the consumer and energy contractor's satisfaction.

In all the sizing methods, the SAO approach appears to be cheapest in terms of levelized cost of energy production. The

Table 13
Summary of the results obtained by the algorithms.

	Sizing results	Algorithms						
		ABC	CSA	GA	PSO	OFA	SAO	AOA
PV/WT/BT	N_{PV}	19	199	58	88	40	210	131
	N_{WT}	67	147	44	53	48	5	59
	N_{Batt}	99	40	56	78	52	44	32
	$N_{Conv/Inv}$	4	4	4	4	4	4	4
	PV cost ($)	608.54	6373.70	1857.66	2818.52	1281.10	6726.00	4195.75
	WT cost ($)	7952.27	17,448.00	5222.39	6290.60	4510.20	593.45	7002.74
	Battery cost ($)	4673.18	1888.20	2643.41	3681.90	2454.60	2077.00	1510.52
	Conv/Inv cost ($)	1130.02	1130.02	1130.02	1130.02	1130.02	1130.02	1130.02
	Total annual cost ($)	14,364.01	26,839.00	10,853.47	13,921.04	9376.00	10,526.44	13,839.00
PV/BT	N_{PV}	65	293	160	72	73	187	96
	N_{WT}	–	–	–	–	–	–	–
	N_{Batt}	114	21	73	66	76	25	147
	$N_{Conv/Inv}$	3	3	3	3	3	3	3
	PV cost ($)	2081.86	9384.38	5124.57	2306.06	2338.10	5989.35	3074.74
	WT cost ($)	–	–	–	–	–	–	–
	Battery cost ($)	5381.23	991.28	3445.88	3115.45	3587.50	1180.09	6938.96
	Conv/Inv cost ($)	847.51	847.51	847.51	847.51	847.51	847.51	847.51
	Total annual cost ($)	8310.61	11,505.68	9417.96	6269.02	6773.10	8016.96	10,861.22
WT/BT	N_{PV}	–	–	–	–	–	–	–
	N_{WT}	12	70	21	68	55	25	15
	N_{Batt}	115	138	66	51	59	105	288
	$N_{Conv/Inv}$	3	3	3	3	3	3	3
	PV cost ($)	–	–	–	–	–	–	–
	WT cost ($)	1424.29	8308.34	2492.50	8070.96	6528.00	2967.24	1780.36
	Battery cost ($)	5428.44	6514.12	3115.45	2407.39	2785.00	4956.40	10,762.46
	Conv/Inv cost ($)	847.51	847.51	847.51	847.51	847.51	847.51	847.51
	Total annual cost ($)	7700.24	15,952.48	6455.47	11,325.87	10,161.00	8771.18	13,390.33

Table 14
The mean, standard deviation, best and worst performances of the algorithms.

Hybrid system	Index	Algorithm						
		ABC	CSA	GA	PSO	OFA	SAO	AOA
PV/wind/battery	Avg	27,834.86	38,073.20	18,074.94	24,578.57	22,891.00	16,790.94	31,107.81
	Best	14,364.01	26,839.00	10,853.47	13,921.03	9376.00	10,526.44	13,839.00
	Std	8588.09	7073.32	9120.59	6339.11	10,667.37	8077.43	13,721.94
	Rank	6	7	3	4	1	2	5
PV/battery	Avg	12,036.71	16,582.61	11,203.39	12,192.99	12,569.50	15,155.91	14,407.80
	Best	8310.61	11,505.68	9417.97	6269.02	6773.10	8016.76	10,861.22
	Std	3529.81	3563.55	1961.75	2964.48	3635.90	3020.76	5189.00
	Rank	5	7	4	1	2	3	6
Wind/battery	Avg	19,479.50	31,640.52	18,717.20	21,016.98	23,399.20	19,439.23	26,730.78
	Best	7700.24	15,952.48	6455.47	11,325.87	10,161.00	8771.18	13,390.33
	Std	7166.62	7860.11	6057.28	7756.58	6931.09	8106.42	8042.10
	Rank	2	7	1	5	4	3	6
Average rank		4.33	7	2.67	3.33	2.33	2.67	5.67
Final rank		5	7	2	4	1	2	6

Table 15
Reliability and economic analysis of hybrid system.

Hybrid systems	Economic parameters	ABC	CSA	GA	PSO	OFA	SAO	AOA
PV/Wind/Battery	LPSP	0.0214	0.0342	0.0229	0.0256	0.0257	0.0090	0.0261
	LCE ($/kWh)	2.21	3.02	1.51660	1.83	1.82	0.06	0.13
	$E_{e/d}$ (kW)	207.48	176.10	556.32	243.93	255.28	431.27	380.08
PV/Battery	LPSP	0.0206	0.0385	0.0136	0.0196	0.0104	0.0196	0.0249
	LCE ($/kWh)	0.96	1.32	0.89	0.97	1.00	0.04	0.0930
	$E_{e/d}$ (kW)	270.97	537.01	270.97	270.49	347.26	393.13	284.25
Wind/Battery	LPSP	0.0017	0.0375	0.0016	0.0033	0.0016	0.0016	0.0016
	LCE ($/kWh)	1.55	2.51	1.49	1.67	1.86	0.052	0.831
	$E_{e/d}$ (kW)	111.48	241.67	343.02	410.44	421.43	619.99	553.47

LPSP also showed that the SAO guaranteed more stable energy for PV/Wind/Battery, Wind/Battery systems. For PV/Battery system, only OFA and GA have better LPSP than the SAO. The average Excess/Deficit energy Computed for each sizing methods also showed that, all the sizing approaches generated an excess of energy for all the hybrid configurations systems. The occurrence of excess/deficit Energy could be considered as an undesirable situation in the hybrid system design, as the energy is wasted. The use of dump load could be considered to address this wastage. However, considering the dump load is beyond the focus of this paper. The cost breakdown of the algorithms for all configurations of the hybrid system are shown in Figs. 9–14. From the figures,

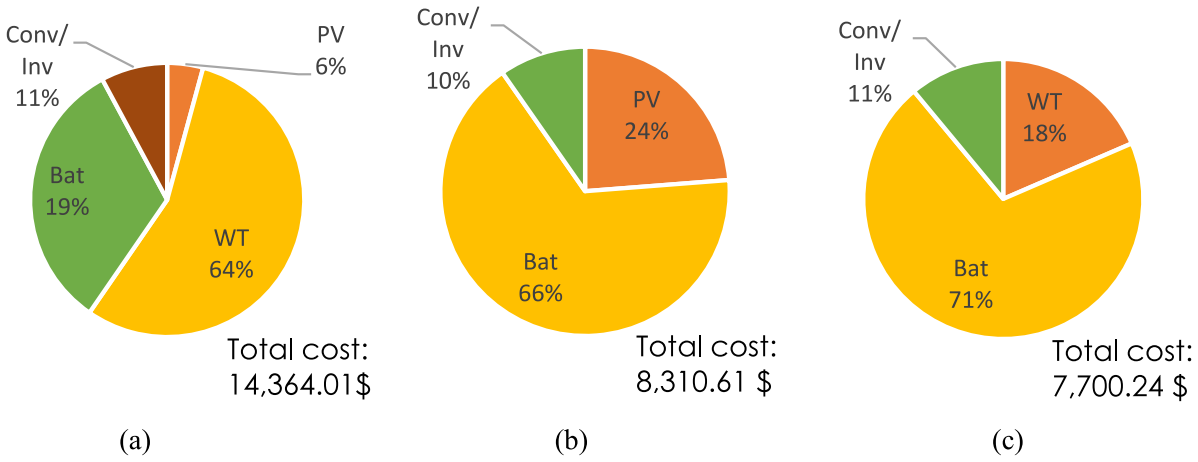

Fig. 9. Break down of the total annual cost by ABC (a) PV/Wind/Battery hybrid system; (b) PV/Battery system; and (c) Wind/Battery system.

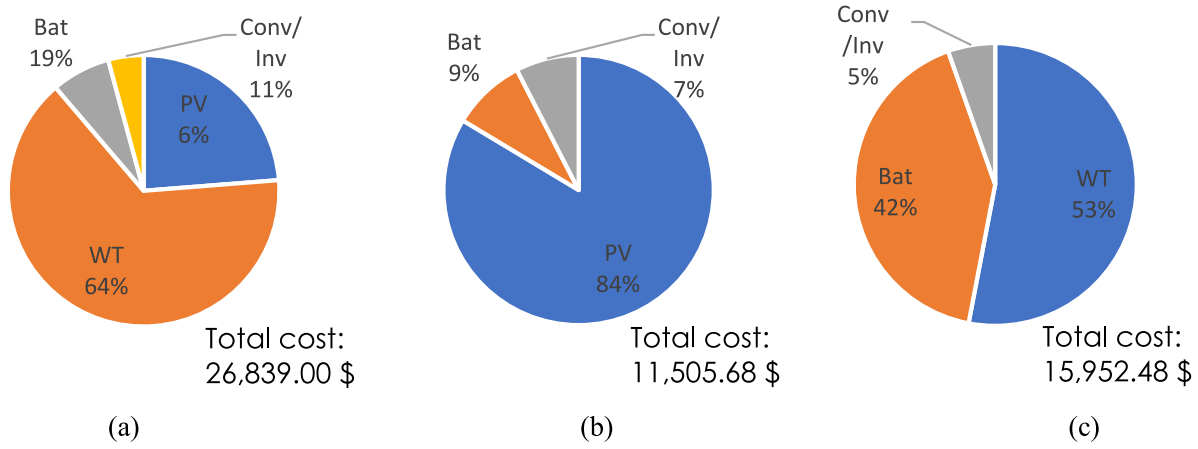

Fig. 10. Break down of the total annual cost by CSA (a) PV/Wind/Battery hybrid system; (b) PV/Battery system; and (c) Wind/Battery system.

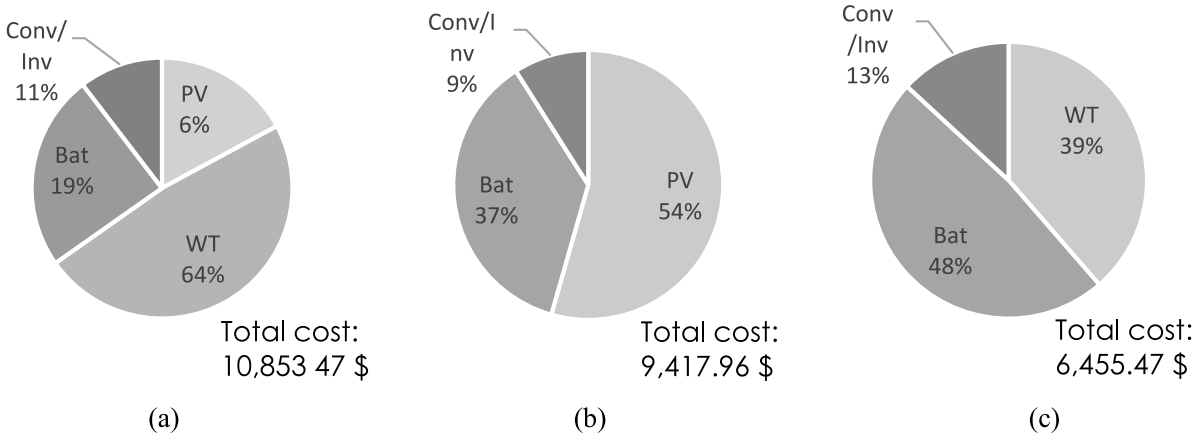

Fig. 11. Break down of the total annual cost by GA (a) PV/Wind/Battery hybrid system; (b) PV/Battery system; and (c) Wind/Battery system.

the average components cost of PV/Wind/Battery, PV/Battery and Wind/Battery configurations are labelled a, b and c respectively. The essence of these plots is to show which of the hybrid components has required higher cost. This information will help an investor during installations (see Fig. 15).

6. Conclusion and recommendation

In this paper, we presented a new metaheuristic algorithm called the smell agent optimization (SAO). The concept of SAO was inspired by the interactions between a smell agent and an object evaporating a smell molecule. This interaction is simulated into three modes namely: sniffing, trailing and random modes in the SAO. Each mode is mathematically simple to implement and requires little programming effort. The performance of SAO was tested on thirty-seven commonly used CEC benchmark functions with diverse properties and sizing of Hybrid Renewable Energy System (HRES) engineering design problem. Statistical results analysis showed that SAO precisely obtained the global

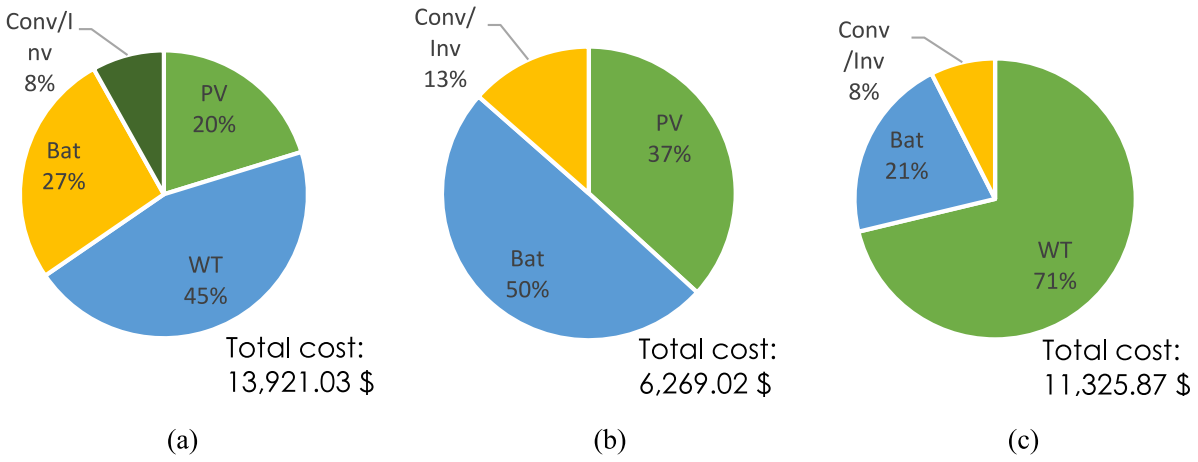

Fig. 12. Break down of the total annual cost by PSO (a) PV/Wind/Battery hybrid system; (b) PV/Battery system; and (c) Wind/Battery system.

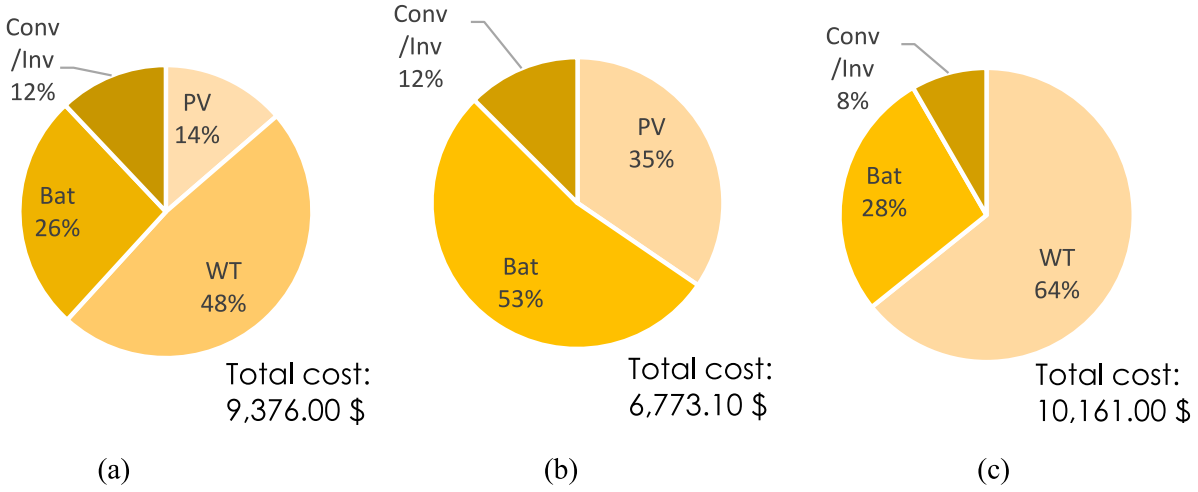

Fig. 13. Break down of the total annual cost by OFA (a) PV/Wind/Battery hybrid system; (b) PV/Battery system; and (c) Wind/Battery system.

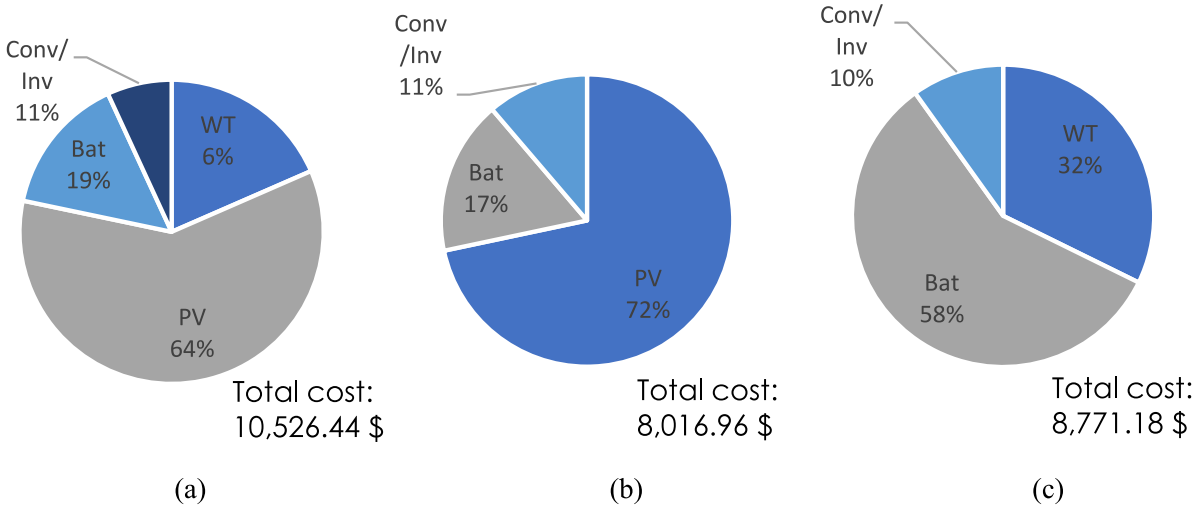

Fig. 14. Break down of the total annual cost by SAO (a) PV/Wind/Battery hybrid system; (b) PV/Battery system; and (c) Wind/Battery system.

solution to 28 of the 37 CEC benchmark functions, surpassing the performance of ABC, CSA, GA, PSO, OFA and AOA which obtained the global solutions in 12, 5, 11, 11, 6 and 15 functions respectively. The SAO demonstrated a faster convergence in most of the benchmark functions by obtaining its optimum solution in fewer function evaluations. The SAO was also tested on HRES engineering optimization problem. Results demonstrated SAO was able to optimize the HRES model effectively more than the benchmark algorithms. The economic analysis of the hybrid systems also showed that SAO achieved a better result considering end to end

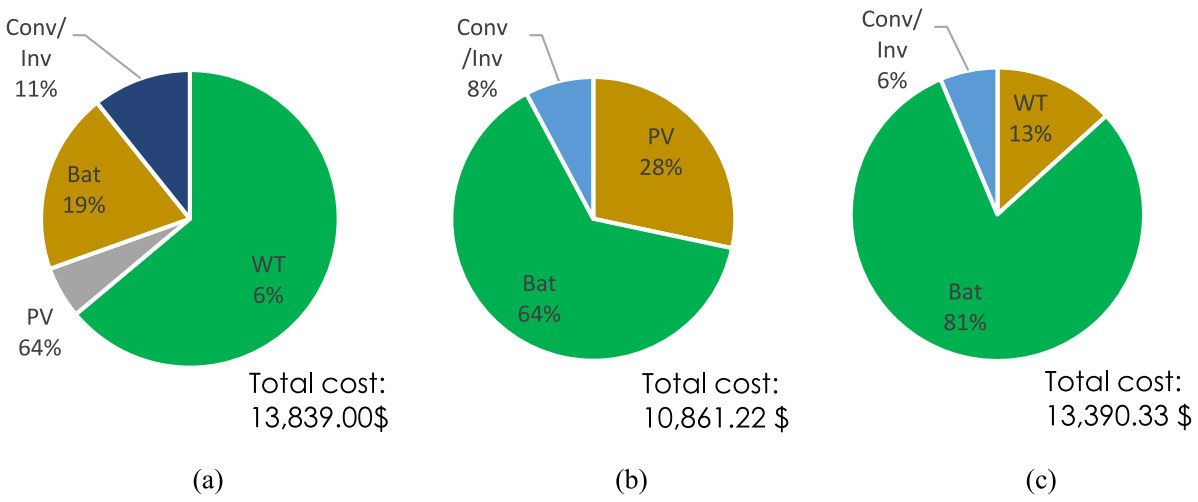

Fig. 15. Break down of the total annual cost by AOA (a) PV/Wind/Battery hybrid system; (b) PV/Battery system; and (c) Wind/Battery system.

satisfactions. The SAO is relatively easy to codify as all the modes require only simple solution search mathematical operations. Unlike most algorithms, SAO has three solution search operation and has a mechanism to escape getting trapped in local optimum. The SAO only has two major control parameters which may further improve the performance of the algorithm if adequately selected and could make the algorithm adaptable to several optimization problems.

As a recommendation, the SAO can be applied to various applications such as feature selection, neural networks, control systems design, power systems, image classification and segmentation, task scheduling, transportation problems, network flow problems, speed reducer design problems, CNN architecture design, data mining, data clustering, wireless sensor networks, etc. For other areas where SAO can be applied, consider the following references [85–88]. Moreover, SAO's can be improved hybridizing the algorithm with good features of other meta-heuristics algorithm. The algorithm can also be combined with levy flight random work, pareto distribution, mutation operators, opposition-based learning, other local and global search methods. Discrete version of the algorithm can also be developed to solve various discrete and binary optimization problems. Performance study of the SOA can be further carried by comparing with another algorithm such as Sine Cosine Algorithm, Symbiotic Organism Search, Cuckoo Search, Arithmetic Optimization Algorithm, Shark Smell Optimization, etc.

The MATLAB implementation of the algorithm can be found in the links provided in Appendix B.

CRediT authorship contribution statement

Ahmed T. Salawudeen: Conceptualization, Methodology, Writing – original draft, Software. **Muhammed B. Mu'azu:** Conceptualization, Supervision, Writing – review & editing. **Yusuf A. Sha'aban:** Supervision, Validation, Writing – review & editing. **Adewale E. Adedokun:** Supervision, Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors will like to appreciate the contributions of Control and Computer research group of the Department of Computer Engineering, Ahmadu Bello University, Zaria. We also extend our gratitude to the unanimous reviewers for their positive comments.

Appendix A. Load data collection questionnaire

Name of Respondent_____
Age_____Sex_____
Specialization_____
Residency _____
Load Data Collection Questionnaire

S/N.	Equipment type	Make	Power rating	Quantity	Hours of operation
1.	Air-conditioner				
2.	Refrigerator				
3.	Microwaves Ovens				
4.	Washing Machine				
5.	Dryer				
6.	TV				
7.	Computers				
8.	Telephones				
9.	DVD player				
10.	Stereo				
11.	Home theatre				
12.	Gaming Console				
13.	Lightening Point				
14.	Printers				
15.	Freezers				
16.	Iron				
17.	Dish Washer				
18.	Fans				
19.	Water Pump				
20.	Electric Cocker				
21.	Water Heater				
22.	Boiler				
23.	Electric Blender				

Appendix B

Implementation of SAO algorithm can be found in the following links:

MATLAB Implementation: https://www.mathworks.com/matlabcentral/fileexchange/99389-smell-agent-optimization

References

[1] A.T. Salawudeen, B.M. Mu'azu, Y.A. Sha'aban, C.J. Chan, Optimal design of PID controller for deep space antenna positioning using weighted cultural artificial fish swarm algorithm, J. Electr. Electron. Syst. 6 (4) (2017).

[2] A.T. Salawudeen, et al., Recent metaheuristics analysis of path planning optimaztion problems, in: 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), Vol. 1, IEEE, Lagos Nigeria, 2020, pp. 1–7.

[3] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.

[4] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A.H. Gandomi, The arithmetic optimization algorithm, J. Comput. Methods Appl. Mech. Eng. 376 (2021) 113609.

[5] S.J. Nanda, G. Panda, A survey on nature inspired metaheuristic algorithms for partitional clustering, Swarm Evol. Comput. 16 (2014) 1–18.

[6] K. Sörensen, Metaheuristics—the metaphor exposed, Int. Trans. Oper. Res. 22 (1) (2015) 3–18.

[7] H. Zhou, M. Song, W. Pedrycz, A comparative study of improved GA and PSO in solving multiple traveling salesmen problem, Appl. Soft Comput. 64 (2018) 564–580.

[8] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.

[9] M. Basu, Modified particle swarm optimization for nonconvex economic dispatch problems, Int. J. Electr. Power Energy Syst. 69 (2015) 304–312.

[10] N. Norouzi, M. Sadegh-Amalnick, R. Tavakkoli-Moghaddam, Modified particle swarm optimization in a time-dependent vehicle routing problem: minimizing fuel consumption, Optim. Lett. (2016) 1–14.

[11] Y. Delice, E.K. Aydoğan, U. Özcan, M.S. Ilkay, A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing, J. Intell. Manuf. 28 (1) (2017) 23–36.

[12] X. Li, J.J.J.o.c. Qian, systems, Studies on artificial fish swarm optimization algorithm based on decomposition and coordination techniques, Vol. 1 (2003) pp. 1–6.

[13] A.T. Salawudeen, Development of an Improved Cultural Artificial Fish Swarm Algorithm with Crossover, Department of Electrical and Computer Engineering, Ahmadu Bello University Zaria, Nigeria. Kubani, 2015, 25.

[14] X.-Y. Luan, Z.-P. Li, T.-Z. Liu, A novel attribute reduction algorithm based on rough set and improved artificial fish swarm algorithm, Neurocomputing 174 (2016) 522–529.

[15] S. Xian, J. Zhang, Y. Xiao, J. Pang, A novel fuzzy time series forecasting method based on the improved artificial fish swarm optimization algorithm, Soft Comput. (2017) 1–11.

[16] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-tr06, Erciyes university, engineering faculty, computer …, 2005.

[17] D. Teodorovic, P. Lucic, G. Markovic, M. Dell'Orco, Bee colony optimization: principles and applications, in: 2006 8th Seminar on Neural Network Applications in Electrical Engineering, IEEE, 2006, pp. 151–156.

[18] X. Chen, X. Wei, G. Yang, W. Du, Fireworks explosion based artificial bee colony for numerical optimization, Knowl.-Based Syst. 188 (2020) 105002, /01/05/ 2020.

[19] J. Yang, J. Cui, Y.-D. Zhang, Artificial bee colony algorithm with adaptive covariance matrix for hearing loss detection, Knowl.-Based Syst. 216 (2021) 106792, 2021/03/15.

[20] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.

[21] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Ieee, 2009, pp. 210–214.

[22] K.M. Passino, Bacterial foraging optimization, Int. J. Swarm Intell. Res. 1 (1) (2010) 1–16.

[23] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspired Comput. 2 (2) (2010) 78–84.

[24] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, 2010, pp. 65–74.

[25] V.S.S. Chandra, Smell detection agent based optimization algorithm, J. Inst. Eng. (India): Ser. B J. Artic. 97 (3) (2016) 431–436.

[26] G.-Y. Zhu, W.-B. Zhang, Optimal foraging algorithm for global optimization, Appl. Soft Comput. 51 (2017) 294–313.

[27] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm, Comput. Struct. 169 (2016) 1–12.

[28] H. Yapici, N.J.A.s.c. Cetinkaya, A new meta-heuristic optimizer: pathfinder algorithm, Vol. 78 (2019) pp. 545–568.

[29] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191, /12/01/ 2017.

[30] L. Abualigah, D. Yousri, M. Abd Elaziz, A.A. Ewees, M.A. Al-qaness, A.H. Gandomi, Aquila optimizer: A novel meta-heuristic optimization algorithm, J. Comput. Ind. Eng. 157 (2021) 107250.

[31] A. Amjadian, A. Gharaei, An integrated reliable five-level closed-loop supply chain with multi-stage products under quality control and green policies: generalised outer approximation with exact penalty, Int. J. Syst. Sci.: Oper. Logist. (2021) 1–21.

[32] C. Duan, C. Deng, A. Gharaei, J. Wu, B. Wang, Selective maintenance scheduling under stochastic maintenance quality with multiple maintenance actions, Int. J. Prod. Res. 56 (23) (2018) 7160–7178.

[33] A. Gharaei, S.A. Hoseini Shekarabi, M. Karimi, E. Pourjavad, A. Amjadian, An integrated stochastic EPQ model under quality and green policies: Generalised cross decomposition under the separability approach, Int. J. Syst. Sci.: Oper. Logist. 8 (2) (2021) 119–131.

[34] A. Gharaei, M. Karimi, S.A.H. Shekarabi, An integrated multi-product, multi-buyer supply chain under penalty, green, and quality control polices and a vendor managed inventory with consignment stock agreement: the outer approximation with equality relaxation and augmented penalty algorithm, Appl. Math. Model. 69 (2019) 223–254.

[35] Y. Hao, P. Helo, A. Shamsuzzoha, Virtual factory system design and implementation: Integrated sustainable manufacturing, Int. J. Syst. Sci.: Oper. Logist. 5 (2) (2018) 116–132.

[36] A. Gharaei, A. Amjadian, A. Shavandi, An integrated reliable four-level supply chain with multi-stage products under shortage and stochastic constraints, Int. J. Syst. Sci.: Oper. Logist. (2021) 1–22, In this issue.

[37] A. Gharaei, B. Naderi, M.J.M.S.L. Mohammadi, Optimization of rewards in single machine scheduling in the rewards-driven systems, 5 (6) (2015) 629–638.

[38] J. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.

[39] R. Storn, K.J.J.o.g.o. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, 11 (4) (1997) 341–359.

[40] R.G. Reynolds, B. Peng, Cultural algorithms: modeling of how cultures learn to solve problems, in: 16th IEEE International Conference on Tools with Artificial Intelligence, IEEE, 2004, pp. 166–172.

[41] S.J.K.-b.s. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Vol. 96 (2016) pp. 120–133.

[42] A. Hatamlou, Black hole: A new heuristic optimization approach for data clustering, Inform. Sci. 222 (2013) 175–184.

[43] S. Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98.

[44] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.

[45] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.

[46] A.R. Mehrabian, C.J.E.i. Lucas, A novel numerical optimization algorithm inspired from weed colonization, 1 (4) (2006) 355–366.

[47] M. Bahrami, O. Bozorg-Haddad, X. Chu, Cat swarm optimization (CSO) algorithm, in: Advanced Optimization By Nature-Inspired Algorithms, Springer, 2018, pp. 9–18.

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

[49] J.-S. Chou, D.-N. Truong, A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean, Appl. Math. Comput. 389 (2021) 125535.

[50] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.

[51] M. Azizi, Atomic orbital search: A novel metaheuristic algorithm, Appl. Math. Model. 93 (2021) 657–683.

[52] F. Zitouni, S. Harous, R.J.I.A Maamri, The Solar System Algorithm: a novel metaheuristic method for global optimization, Vol. 9 (2020) pp. 4542–4565.

[53] E. Bogar, S.J.A.S.C. Beyhan, Adolescent Identity Search Algorithm (AISA): A novel metaheuristic approach for solving optimization problems, Vol. 95, 2020, p. 106503.

[54] S.S.V. Chandra, Smell detection agent based optimization algorithm, J. Inst. Eng.: Ser. B 97 (3) (2016) 431–436.

[55] L.B. Buck, Unraveling the sense of smell (nobel lecture), Wiley Online Libr. 44 (38) (2005) 6128–6140.

[56] E. Sakalli, D. Temirbekov, E. Bayri, E.E. Alis, S.C. Erdurak, M. Bayraktaroglu, Ear nose throat-related symptoms with a focus on loss of smell and/or taste in COVID-19 patients, American Journal of Otolaryngology 102622 (2020) 102622.

[57] R. Axel, Scents and sensibility: a molecular logic of olfactory perception (nobel lecture), in: Wiley Online Library, Vol. 44, Angewandte Chemie International ed., Wiley Online Library, 2005, pp. 6110–6127, no. 38.
[58] H. Bayir, et al., Evaluation of olfactory memory after sevoflurane anesthesia: is really shortterm memory influenced?, 2016.
[59] S. Sookoian, A. Burgueño, T.F. Gianotti, G. Marillet, C.J. Pirola, Odor perception between heterosexual partners: its association with depression, anxiety, and genetic variation in odorant receptor OR7d4, Biological Psychology 86 (3) (2011) 153–157.
[60] J.E. Amoore, E. Hautala, Odor as an ald to chemical safety: odor thresholds compared with threshold limit values and volatilities for 214 industrial chemicals in air and water dilution, J. Appl. Toxicol. 3 (6) (1983) 272–290, Wiley Online Library.
[61] R. Stevenson, An initial evaluation of the functions of human olfaction, Chem. Senses 35 (1) (2010) 3–20.
[62] S. Menzel, T. Hummel, L. Schäfer, C. Hummel, I. Croy, Olfactory change detection, Biol. Psychol. 140 (2019) 75–80.
[63] A.T. Salawudeen, M.B. Mu'azu, A. Yusuf, E.A. Adedokun, From smell phenomenon to smell agent optimization (SAO): A feasibility study, in: International Conference on Global & Emerging Trends, Vol. 1, Global Trends Academy, Abuja Nigeria, 2018, pp. 79–85, no. 2.
[64] S. Chapman, T.G. Cowling, D. Burnett, The Mathematical Theory of Non-Uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases, Cambridge University Press, 1990.
[65] M. Abdechiri, M.R. Meybodi, H. Bahrami, Gases Brownian motion optimization: an algorithm for optimization (GBMO), Appl. Soft Comput. 13 (5) (2013) 2932–2946.
[66] F. Rottstaedt, et al., Size matters–the olfactory bulb as a marker for depression, J. Affect. Disord. 229 (2018) 193–198.
[67] E. Çelik, A powerful variant of symbiotic organisms search algorithm for global optimization, Eng. Appl. Artif. Intell. 87 (2020) 103294.
[68] S. Saha, V. Mukherjee, A novel chaos-integrated symbiotic organisms search algorithm for global optimization, Soft Comput. 22 (11) (2018) 3797–3816.
[69] K.H. Truong, P. Nallagownden, Z. Baharudin, D.N. Vo, A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems, Appl. Soft Comput. 77 (2019) 567–583.
[70] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inform. Sci. 192 (2012) 120–142.
[71] A. Ogunjuyigbe, T. Ayodele, O. Akinola, Optimal allocation and sizing of PV/Wind/Split-diesel/Battery hybrid energy system for minimizing life cycle cost, carbon emission and dump energy of remote residential building, Appl. Energy 171 (2016) 153–171.
[72] J. Yu, J.-H. Ryu, I.-b. Lee, A stochastic optimization approach to the design and operation planning of a hybrid renewable energy system, Appl. Energy 247 (2019) 212–220, /08/01/2019.
[73] A.S. Al Busaidi, H.A. Kazem, A.H. Al-Badi, M. Farooq Khan, A review of optimum sizing of hybrid PV–wind renewable energy systems in oman, Renew. Sustain. Energy Rev. 53 (2016) 185–193, 01/01/2016.
[74] L. Bartolucci, S. Cordiner, V. Mulone, S. Pasquale, Fuel cell based hybrid renewable energy systems for off-grid telecom stations: Data analysis and system optimization, Appl. Energy 252 (2019) 113386, 2019/10/15.
[75] D. Mazzeo, N. Matera, P. De Luca, C. Baglivo, P. Maria Congedo, G. Oliveti, Worldwide geographical mapping and optimization of stand-alone and grid-connected hybrid renewable system techno-economic performance across Köppen-Geiger climates, Appl. Energy 276 (2020) 115507, 2020/10/15.
[76] S. Cordiner, et al., Fuel cell based hybrid renewable energy systems for off-grid telecom stations: Data analysis from on field demonstration tests, Appl. Energy 192 (2017) 508–518, 04/15/2017.
[77] I. Vázquez-Fernández, M. Raghibi, A. Bouzina, L. Timperman, J. Bigarré, M. Anouti, Protic ionic liquids/poly(vinylidene fluoride) composite membranes for fuel cell application, J. Energy Chem. 53 (2021) 197–207, 02/01/2021.
[78] F.J. Vivas, A. De las Heras, F. Segura, J.M. Andújar, A review of energy management strategies for renewable hybrid energy systems with hydrogen backup, Renew. Sustain. Energy Rev. 82 (2018) 126–155, 02/01/2018.
[79] X. Li, W. Wang, H. Wang, J. Wu, X. Fan, Q. Xu, Dynamic environmental economic dispatch of hybrid renewable energy systems based on tradable green certificates, Energy 193 (2020) 116699, 2020/02/15.
[80] J. Liu, M. Wang, J. Peng, X. Chen, S. Cao, H. Yang, Techno-economic design optimization of hybrid renewable energy applications for high-rise residential buildings, Energy Convers. Manage. 213 (2020) 112868, 2020/06/01.
[81] Y. Sawle, S.C. Gupta, A.K. Bohre, Socio-techno-economic design of hybrid renewable energy system using optimization techniques, Renew. Energy 119 (2018) 459–472, 04/01/2018.
[82] M.S. Javed, D. Zhong, T. Ma, A. Song, S. Ahmed, Hybrid pumped hydro and battery storage for renewable energy based power supply system, Appl. Energy 257 (2020) 114026, 2020/01/01.
[83] K. Li, S. Zhang, Y. Li, J. Fan, K. Lv, Mxenes as noble-metal-alternative co-catalysts in photocatalysis, Chinese J. Catal. 42 (1) (2021) 3–14, 01/01/2021.
[84] M.A.M. Khan, S. Rehman, F.A. Al-Sulaiman, A hybrid renewable energy system as a potential energy source for water desalination using reverse osmosis: A review, Renew. Sustain. Energy Rev. 97 (2018) 456–477, 12/01/2018.
[85] A. Gharaei, S.A. Hoseini Shekarabi, M. Karimi, Modelling and optimal lot-sizing of the replenishments in constrained, multi-product and bi-objective EPQ models with defective products: Generalised cross decomposition, Int. J. Syst. Sci.: Oper. Logist. 7 (3) (2020) 262–274.
[86] A. Gharaei, M. Karimi, S.A. Hoseini Shekarabi, Joint economic lot-sizing in multi-product multi-level integrated supply chains: Generalized benders decomposition, Int. J. Syst. Sci.: Oper. Logist. 7 (4) (2020) 309–325.
[87] S.A. Hoseini Shekarabi, A. Gharaei, M. Karimi, Modelling and optimal lot-sizing of integrated multi-level multi-wholesaler supply chains under the shortage and limited warehouse space: generalised outer approximation, Int. J. Syst. Sci.: Oper. Logist. 6 (3) (2019) 237–257.
[88] R. Sayyadi, A. Awasthi, A simulation-based optimisation approach for identifying key determinants for sustainable transportation planning, Int. J. Syst. Sci.: Oper. Logist. 5 (2) (2018) 161–174.