Empirical Study of Sperm Swarm Optimization Algorithm

Chapter · January 2019

CITATION
1

READS
251

3 authors:

Hisham Shehadeh
Amman Arab University
19 PUBLICATIONS **75** CITATIONS
SEE PROFILE

Ismail Ahmedy
University of Malaya
70 PUBLICATIONS **577** CITATIONS
SEE PROFILE

Mohd Yamani Idna Idris
University of Malaya
140 PUBLICATIONS **1,845** CITATIONS
SEE PROFILE

Empirical Study of Sperm Swarm Optimization Algorithm

Hisham A. Shehadeh, Ismail Ahmedy(✉), and Mohd Yamani Idna Idris(✉)

Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
sh7adehl990@hotmail.com, {ismailahmedy,yamani}@um.edu.my

Abstract. This paper gives an empirical study to estimate the performance of our proposed optimization method called Sperm Swarm Optimization (SSO). The SSO is evaluated frequently with different mathematical benchmark models utilized in the scope of optimization. Various asymmetric parameters and settings are chosen for these benchmark functions. The acquired results are compared with the results of four methods, such as Genetic Algorithms (GA), Parallel Genetic Algorithm (PGA), Particle Swarm Optimization (PSO) and Accelerated Particle Swarm Optimization (APSO). The outcomes present that the proposed approach outperforms other approaches in terms of quality of result because of using the technique of inherently continuous to update the sperm location. In addition, it uses different types of mutations which are utilized to increase the method convergence.

Keywords: Optimization · Evolutionary computation · Meta-heuristic Optimal · Algorithms

1 Introduction

Recently, the use of optimization algorithms has been rising with increasing the complexity of real-life problems. These methods can be classified in various ways, which can be mainly categorized into two groups: stochastic methods, and deterministic methods. The former always has a step of generating slightly various individuals (solutions) at each round of procedure, while the latter have rigorous steps in their predefined procedures, which can be achieved some solutions if the procedures iterations begin from the generating of initial solutions.

Mainly, deterministic and stochastic meta-heuristic methods have been used to solve a wide range of complicated and complex non-linear optimization problems such as permutation flow shop scheduling [1], reliability-robust design optimization problems [2], education composition [3], water and geotechnical engineering [4, 5], problems of mobile networks [6], and engineering designs [7].

Optimization algorithms and techniques have been created; depends on a metaphor of some manmade or natural process. The main target is to obtain a solution that is at least near to an optimal solution. In practice, optimization algorithms can often archive

K. Arai et al. (Eds.): IntelliSys 2018, AISC 869, pp. 1082–1104, 2019.
https://doi.org/10.1007/ 978-3-030-01057-7_80

optimal solution by using a population of solutions (individuals), though there is no ability to obtain the global optimal solution. Since the creation of GA [8] as an efficient optimization approach, after that, different meta-heuristic optimization methods have been proposed, such as Differential Evolution (DE) [9], Cuckoo Search (CS) [10], Genetic Programming (GP) [11], Ant Colony Optimization (ACO) [12], Artificial Bee Colony (ABC) [13], PSO [14], APSO [15], PGA [16], etc. On the other hand, Sperm Swarm Optimization (SSO) is inspired by motility of sperm to fertilize the Ovum (egg). Nevertheless, they all operate in the same way that is, initializing and then updating each individual in the population by performing some types of operators according to the fitness information acquired from their environment so that each individual in the population can be expected to move towards areas of better solution.

The first work on SSO algorithm was on optimizing some problems in the area of Wireless Sensor Network (WSN) [17]. However, the work did not cover the topic of optimizing standard benchmark models. The main objectives of this paper are to test this algorithm with different benchmark functions and evaluate them by making a comparison with other optimization approaches such as APSO, PSO, GA, and PGA. We organized our paper as follows: Literature review is summarized in Sect. 2. Section 3 summarizes sperm swarm optimization algorithm. Section 4 shows experimental results. This work is concluded in Sect. 5.

2 Literature Review

This part of work provides a general review related to meta-heuristic optimization algorithms. It also gives the need for the work and presents view in-depth at different relevant kinds of literature on meta-heuristic approaches, which are implemented in the present and past to optimize different types of problems. Figure 1 shows a statistical report about the optimization algorithms used in the publications stored in ISI Web of Knowledge and IEEE Xplore databases [18]. As we can notice from the chart, GA and PSO algorithms are the mostly used methods to solve different kinds of complicated non-linear optimization problems [18]. For this reason, we chose GA, PGA, PSO, and APSO to compare them with our proposed algorithms. In addition, this section highlights the advantages with drawbacks and comparison between both PSO and GA.

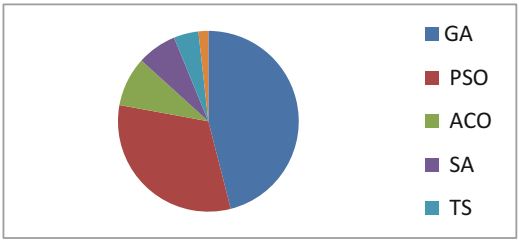

Fig. 1. A statistical report about the optimization algorithms used in the publications stored in ISI Web of Knowledge databases and IEEE Xplore [18].

2.1 Genetic Algorithm (GA)

Genetic algorithm is considered as the pioneer approach in the area of optimization. This algorithm is proposed by Holland in 1993 [19]. GAs are the collection of adaptive procedures and operations, which are used to give a solution to the optimization and search problems by exploring the optimal solution in large-scale search space [20]. This algorithm simulates a natural procedure of choosing the most suitable chromosome in a set of the population to offer an optimal or near from optimal solution to different kind of problems [20]. GA can also be referred to Biology Inspired Search (BIS) method that works using a procedure closer in behavior to the theory of evolution that discovered by Charles Darwin [21]. Based on Goldberg research, GA has been considered as an algorithm that can find a solution for complex problems that many kinds of classical algorithms cannot provide suitable answers for these problems in a short time [22, 23].

Depending on the procedure of GA, an initial population is arbitrarily created, and each solution is evaluated. After that, many solutions with higher fitness are reserved for next step, while the solutions with lower fitness are throwing out of the population. The GA uses natural genetic operators and factors such as natural selection, crossover with the mutation to create better generation [24, 25]. The aim of crossover is to select good chromosomes in hope to produce an excellent offspring (new solutions). In addition, mutation is a procedure of any change on genes that can affect the result [26]. The Genetic Algorithm (GA) pseudo-code is summarized in Algorithm 1 [26].

Algorithm 1 *Genetic Algorithm (GA)*
Begin
Step 1: *Set the problem based on a genetic representation.*
Step 2: *Create an initial population* $p(0) = x_1^0, \ldots x_N^0$.
Step 3: *Calculate the average fitness* $\bar{F} = \sum_{i}^{N} F(x_i)/N$ *and give a normalized fitness value for each individual* $F(x_i^t)/N$.
Step 4: *Give each* x_i *a probability* $p(x_j, t)$ *proportional to its normalized fitness. Based on this distribution, select N vectors from P(t). This provides the set S(t).*
Step 5: *Pair values in S(t) at random forming N/2 pairs. Perform crossover with probability pcross to each pair. Perform mutation to form a new population P(t+1).*
Step 6: *Set t = t +1, go to step 3.*
End.

2.2 Parallel Genetic Algorithm (PGA)

PGA performs two enhancements on the standard GA. In the first modification, selection for mating is distributed, which each individual lives in 2-D search space and the selection of a solution are done independently for each individual based on its neighborhood. In the second modification, each individual has the ability to enhance its

fitness value during its existence by using some techniques such as local hill-climbing, which each individual has the ability to use different local hill-climbing methods. The Parallel Genetic Algorithm (PGA) pseudo-code is summarized in Algorithm 2 [16]. The parallel search can be done by linking the individual with the neighborhood. This linkage can be done probabilistically constrained by the neighborhood. PGA can be considered as a distributed algorithm, which each individual made its own decision without central control.

Algorithm 2 *Parallel Genetic Algorithm (PGA)*
Begin
Step 1: *Set the problem based on a genetic representation.*
Step 2: *Create an initial population and the structure of the population.*
Step 3: *Perform the local hill climbing on each individual.*
Step 4: *Partner for mating has been selected by each individual based on its neighborhood.*
Step 5: *Perform crossover on the parents to produce an offspring set.*
Step 6: *Perform the hill climbing on the offspring*
Step 7: *Go to step 4 if not finished,*
End.

2.3 Particle Swarm Optimization (PSO)

PSO was originally proposed by Kennedy et al. in 1995 [14, 27]. This algorithm is inspired by particles movements, which each particle has location and velocity. Each particle in search space represents a possible solution in PSO, as they explore the problem search space searching for the best or near to the best solution. Particles change their position according to their accumulated data and knowledge of exploring in the searching space, in PSO, (Vi) simulates the velocity of the particle and (Xi) simulates the position of each particle.

i is used to represent the particle number, where (i = 1,…, N), and the maximum number of solutions in the swarm represents as N. The $X_i = (X_{i1,}X_{i2,}X_{i3}, \ldots X_{iN})$ equation is used to represent ith particles. The present velocity of each particle is the degree gained depends on the past location of each particle. The $V_i = (V_{i1,}V_{i2,}V_{i3}, \ldots V_{iN})$ equation is used to represent the velocity of particle i. Through the algorithm, the position on the dimensional search space and velocity of each particle are given numerical values randomly. This step is followed by using a set of equations such as (1) and (2) to update the location and velocity of the particles until reaching the maximum number of search iterations.

$$fV_{i,m}^{(t+1)} = w * v_{i,m}^{(t)} + c_1 * rand_1()* (pbest_{i,m} - x_{i,m}^{(t)}) + c_2 * rand_2() * (gbest_m - x_{i,m}^{(t)}) \quad (1)$$

$$X_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (2)$$

It is very important to test the performance of an algorithm by executing it with different conditions. For this reason, Shi and Eberhart [27] prove that the performance of PSO is not identical in case of different population size. On the other hand, C_1 and C_2 are two learning factors which should be equal and their ranges are between (0, 4) [28]. In addition, there are two random numbers called stochastic variables and they have random ranges between (0, 1). These parameters are applied to increase the convergence of the algorithm. The stochastic variables are stand-alone functions that raise the convergence and velocity of each particle.

There are two ideal locations for the swarm, the first location is the ideal location that located by the particle itself, the second location is the best location that located by the neighboring particles. There are two types of positions that located by neighboring, including, global neighborhood, which is the location that connected to the best particle in the swarm. However, local neighborhood is the position that related only to particle neighbors itself. Equation (2) mentioned above is used to update the particles positions, whereby the earlier position is incremented together with the new velocity. After the positions update, a new search is begun from the former positions. The optimal solution is the ideal position found when the maximum number of iteration or stopping criterion is reached or satisfied [29]. PSO used an inertia weight (inertia parameter), which is represented as w. Eberhart et al. [30] discussed that it is appropriate to begin the search via a larger inertia value, which helps to get more global and local searches. However, using a small value of inertia helps to raise the method convergence rather than exploring the whole global search space [30]. There are two different ways that have been discussed for choosing appropriate values for inertia operator. The first one is a linear method in which the value of method decreases linearly until reaching the maximum available number of inertia [27]. Equation (3) represents linear method.

$$w_{i+1} = w_{\max} - \frac{w_{\max} - w_{\min}}{i_{\max}} i \tag{3}$$

The second one is called a dynamic method in which the value of this factor reduces from the initial value to the determined final value fractionally by (4).

$$w_{i+1} = \Delta_w w_i \tag{4}$$

Where the value of Δ_w varies between 1 and 0. Shi and Eberhart [27], prove that dynamic method outperforms linear method in term of convergence. The PSO pseudo-code is summarized in Algorithm 3 [27].

Algorithm 3 *Particle Swarm Optimization (PSO)*
Begin
Step 1: ***initialize particles.***
Step 2: ***for*** *i=1: total numbers of particles in P do*
Calculate fitness value fp=f(p).
if f_p *is better than f(pbest)*
Set p best=p.
end if
end for
Step 3: Set gbest = choose the best value in population (P).
Step 4: ***for*** *i=1: total numbers of particles in P* ***do***
Calculate velocity for each particle.
end for
Step 5: *Go back to step 2 and repeat until reaching the maximum iteration.*
End.

2.4 Accelerated PSO (APSO)

As we mentioned previously, PSO is an algorithm works to update the location of each particle based on the global best location and individual best location. APSO is created based on eliminating the individual best location and use only the global best location to speed up the algorithm convergence. The velocity update rule of this algorithm is summarized in the following equation [31]:

$$v_i^{t+1} = v_i^t + \alpha r(t) + \beta r(g^* - x_i^t) \tag{5}$$

Where r takes a value in the range of 1 and 0 randomly. The velocity of each individual in APSO is not essential and thus can be eliminated. In order to increase the algorithm convergence, the location of the particle can be written in a single step:

$$x_i^{t+1} = (1-\beta)x_i^t + \beta g^* + \alpha r \tag{6}$$

From the previous equation, we can notice that the equation does not contain the velocity. For this reason, the APSO does not require the initial velocities to update the swarm positions on search space. So, the APSO is more easy and simple to implement. The basic term in APSO position update rule is (r) in which is utilized to perform the swarm mobility. The determination of r value depends on the value of α, which $\alpha = 0.1 - 0.5L$. The scale of each variable is represented by L, while β has the range of 2.0–0.7, which is appropriate for the most real-world problem [31]. There are solutions to reduce the randomness gradually by using the following equation:

$$\alpha = \delta^t \tag{7}$$

Where in most cases δ has the range of 0.1–0.99. $t \in [0, t_{\max}]$ and t_{max} is the maximum value of generations [31]. The full procedure of APSO can be summarized as following steps [15]:

Algorithm 4 *Accelerated particle swarm optimization (APSO)*
Begin
Step 1: *Assign the location of all particles randomly* ***(Initialization)***.
Step 2: *Estimate the fitness of the initial population.*
Step 3: ***While*** *(t<MaxIteration)* ***do***
for *i=1:number of particles* ***do***
Use *Eq.(6) to update the locations of all particles*
Check xi, after that, limit to the allowed limits if appropriate.
end for *i*
Update the counter: t=t+1
Step 4: ***end while***
Step 5: *Print the values of both g*and f(g*).*
End.

2.5 GA Versus PSO

It should be noted that the PSO is approach with inherently continuous, i.e. it applies three phases to evaluate and update the population until reaching the final number of iterations. First, it generates population velocity and position. Second, it updates velocity. Third, it updates position, which is considered as the final step of the PSO procedure. GA is considered as an inherently discrete approach, which is utilized to encode individuals into 0's and 1's; based on that, it easily applies a mode of discrete settings and variables. In GA, the procedure performs the natural selection operations, crossover with mutation operations. In a different view, in PSO the new location of each particle is created based on the prior location, which the global best and the neighborhood locations lead the search on the problem space domain [32, 33].

There are many advantages of GA and PSO. We can summarize a few of them in the following points [34, 35]:

- GA can solve different optimization problems, which can be represented by chromosome encoding.
- The execution technique in GA is not affected by the error surface, which can solve multi-dimensional, parametrical, non-parametrical, continuous, non-continuous and even non-differential problems.
- GA is easy to understand and implement.
- The idea of PSO is conducted from the intelligence. It can be applied into both engineering use and scientific research.
- PSO has no mutation and overlapping calculation.
- The calculation and complexity in PSO are very simple, which can be easily applied.
- The calculation in PSO based on real numbers, which is decided directly by the solution. The dimension value is always equal to the solution, which is a constant value.

There are many drawbacks of the previous algorithms. We can summarize them in the following points:

- In GA, mutation is used to increase the convergence. However, it influences the quality of the solution. Different mutation percentages lead to various solutions [36]. This verity leads to a misunderstanding of the optimality.
- PSO has the issue of failing to discover the local optimum in wide and high-dimensional search space [37] because the exploration ability is reduced to locate at zones, which contain good solutions.
- In PSO, diversity of the learning factor C_1 and C_2 influence the quality of the solution in which different values of C_1 and C_2 lead to various solutions. C_1 and C_2 play a significant role in particle velocity, which if they increase, the velocity will be increased.
- In PSO, the inertia factor values affect the convergence [27, 38], which if the inertia factor is small the particle velocity will be slow.

3 Sperm Swarm Optimization Algorithm

SSO can be deemed as an approach with distributed behavior that can be utilized to solve multi-dimensional problems. SSO is inspired by motility of sperm to fertilize the Ovum (egg), which the sperm move forward in a swarm from a low temperate zone called Cervix. During this direction, sperm searches for a high-temperature zone called Fallopian Tubes where the egg is waiting for the swarm for fertilization in this zone. At the end, there is one sperm which will fertilize the egg. This sperm is called the winner. Figure 2 shows sperm swarm and the winner reaches the egg.

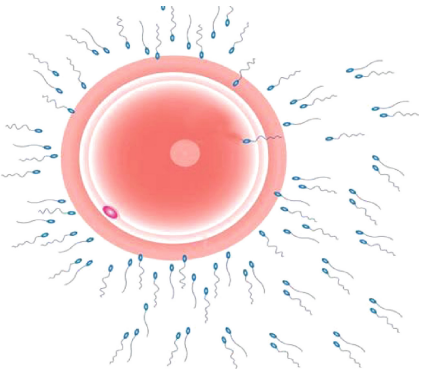

Fig. 2. Sperm swarm and the winner [17, 39].

There are many limitations on sperm movement. We can summarize them as follows:

- pH value inside the reproductive system of female:

(1) The pH value inside the reproductive system of female takes a value in the range of 4.5–5.5 [40], which are represented as normal pH values for a healthy vagina.
(2) However, sperm does not like low pH values, for that reason, in the time of ovulation, the pH value of acidic or acid inside vagina is anywhere stable around 7–14, which is suitable for movement of the sperm and is deemed as alkaline and non-toxic to the swarm [41]. In other words, low pH of mucus acidic may destroy the sperm cell.
(3) Kind of food consumed [42] and mood status or emotion of the female, such as sadness or happiness, etc. [43] play an important role in determining the pH value of mucus acidic inside the female reproductive system. Depends on these facts, we can observe that the pH value will be changed around 7-14.

- Temperature inside the reproductive system of female:

(1) As we aforesaid, sperm head works like temperature sensor, which searches on the warmer area (the egg location). The temperature inside the reproductive system of female can be changed depends on women status.
(2) The researchers found that the temperature average inside the vagina has a value around 35.1 to 37.4 in degrees Celsius [44].
(3) Nevertheless, this temperature may increase until it reaches 38.5 °C in some circumstances because of circulation of vaginal blood pressure [45]. Depending on these facts, we can observe that the value of temperature will be changed around 35.1–38.5.

Depending on the previous information, we can observe that sperm velocity is affected by both of the temperature and pH value inside the vagina, which play an important role in sperm movement and its movement direction. We can summarize the velocity of sperm in the following steps:

Sperm initial velocity is the velocity that gained after the ejaculation task inside the cervix zone. Each sperm gains a random location inside the cervix, which its velocity is affected by the value of pH in this location. The initial velocity can be calculated by the following equation:

$$Initial\_Velocity = D \cdot V_i \cdot Log_{10}(pH\_Rand_1) \tag{8}$$

Where D is a factor of velocity damping, which takes a value between 1 and 0 randomly. This factor is used to adjust the velocity of sperm. V_i represents the sperm velocity. pH_Rand_1 is a number takes a value in the range of 7 to 14 randomly, which represents the pH value.

Personal best solution is the best solution that obtained by the sperm by itself. This solution can be achieved by comparing the past location of the sperm with its current achieved location. This location will be stored in the memory just if the past location is

worse than current location. The personal best solution of the sperm can be calculated by the following equation:

$$\begin{aligned} Current\_Best\_Solution &= Log_{10}(pH\_Rand_2)\cdot \\ Log_{10}(Temp\_Rand_1)&\cdot (sb\_solution[] - current[]) \end{aligned} \tag{9}$$

Where sb_solution [] is the sperm best solution. pH_Rand_2 is a number takes a value in the range of 7 to 14 randomly, which simulates the pH value. $Temp\_Rand_1$ is a number takes a value in the range of 35.1 to 38.5 randomly, which simulates the temperature of the visited area.

Global best solution is the solution of the sperm with the closest location to the optimal solution (location of egg). This value simulates the value of the winner. The global best solution can be calculated by the following equation:

$$\begin{aligned} &Global\_Best\_Solution = Log_{10}(pH\_Rand_3)\cdot \\ &Log_{10}(Temp\_Rand_2)\cdot \\ &(sgb\_solution[] - current[]) \end{aligned} \tag{10}$$

Where sgb_solution[] is the swarm global best solution. pH_Rand_3 is a number takes a value in the range of 7 to 14 randomly, which simulates the pH value. $Temp\_Rand_2$ is a number takes a value in the range of 35.1 to 38.5 randomly, which simulates the temperature of the visited area. current [] is the current best solution, which can be calculated by the following formula:

$$current[] = current[] + v[] \tag{11}$$

Where *v[]* represents the sperm velocity, which can be calculated by using (12).

We can represent the sperm velocity update rule as (12). This rule represents three different velocities that are used by each sperm to reach the location of the egg (the optimal solution), we can summarize these velocities as following steps. The first velocity is the sperm initial velocity, which is acquired after the ejaculation in cervix zone. This velocity is influenced by the pH value in the cervix zone. The second velocity is a personal best solution, which is recorded by the best solution found by the sperm itself. This velocity is influenced by two factors, including, the value of pH and temperature of the visited zone, which zone temperature helps the sperm to know the egg location. The third velocity is the global best solution, which is recorded by the swarm based on the winner solution. As we mentioned previously, there is only one sperm that will fertilize the egg because its solution is the closest to the target. This velocity is affected by two factors, including, the value of pH and temperature of the visited zone. The temperature is a very important factor, which helps the sperm to know the egg location.

$$\begin{aligned} v[] = &Initial\_Velocity + Current\_Best\_Solution \\ &+ Global\_Best\_Solution \end{aligned} \tag{12}$$

From the previous models, we can create a full Sperm Swarm Optimization (SSO) algorithm as follows:

Algorithm 5 *Sperm Swarm Optimization (SSO)*
Begin
Step 1: Initialize locations for sperm swarm.
Step 2: ***for*** *i=1: population size* ***do***
Step 3: *Estimate the fitness for swarm*
if *obtained fitness > best solution of thesperm* ***then***
Assign the current value as the best solution of the sperm
end if
end for
Step 4: *Assign the global best solution depends on the winner.*
Step 5: ***for*** *i=1: population size* ***do***
Perform the velocity update rule (sperm swim)
Update the position of the sperm on the problem search space
end for
Step 6: ***for*** *i=1:population size* ***do***
if (i % 3 == 0) then
Perform a non-uniform mutation
else if (i % 3 == 1)
Perform a uniform mutation
else
Do not apply mutation
end if
end for
Step 7: ***while*** *maximum iterations are not reached go back to 2 and repeatuntil reaching the maximumnumber of itration.*
End.

The logarithm has been applied to both the temperature and pH factors to normalize these parameters to be small values. This leads to achieve a reasonable slow velocity that mimics the normal motility of a real sperm.

In order to enhance the performance and convergence of SSO algorithm, the population size is divided into three equal portions by taking the mod (modulus) of sperm index (population index) on the value of three. In the first portion of the population, the non-uniform mutation operator is performed; in the second portion, a uniform mutation operator is performed. On the other hand, in the third portion, mutation is not performed. This helps to raise the convergence of the method and if the mutation operators do not provide good results, the population without mutation (a third portion of the population) will save on good results.

Depends on the prior equations and rules, it easy to demonstrate that SSO is different than evaluation methods such as GA and PGA in which are approaches with inherently discrete. In addition, evaluation methods use adaptive procedures such as adaptive selection scheme [46], which is easy to implement discrete design parameters and variables. Therefore, the static parameters and variables can be included easily in their operation. In a different view, SSO performs approach of inherently continuous to update the swarm location and velocity, which the past position of each sperm in the swarm will be stored in the memory and depends on it the sperm generates a new position. The random parameters are used in SSO (i.e., pH and temperature), which play an important role in changing the location of the sperm until reaching the optimal solution. The randomness has been used by many well-known optimization algorithms

such as PSO. Examples of these factors are C_1 and C_2, which have value around 0 to 4 randomly.

In SSO, the history of data samples is not stored in the memory for each sperm such as temperature value and the pH value of the visited location. Only, the location of sperm will be cached in memory. The location of each sperm is the outcome of multiplying different numerical parameters with each other's, such as the sperm best solution, temperature value, and pH value, etc. SSO uses the past location (cached location) to compare it with the new location and it replaces the past location by new location just if the new location is better than the past location. This is very clear in the Algorithm 5 and the mathematical formulas from 8 to 12.

4 Experiment and Results

To validate and evaluate the efficiency of the SSO, it is utilized to solve a set of benchmark models, which are summarized in the following points. Benchmark function is a nonlinear function, which is used to evaluate the meta-heuristic methods [47]. We select these functions carefully as they are considered as standard benchmark models for evaluation of single-objective optimization approaches [48–52]. All the listed benchmark functions are minimization problems, which all of their results should be minimum. Furthermore, we compare the results of the proposed approach with PSO, APSO, GA and PGA approaches. The benchmark functions that are used in this work are listed in the following points:

- **Sphere function**: is the nonlinear function, which has a shape like a sphere. This function has a known value of global minimum at (0, 0) with an optimal value of zero. The mathematical description of this function is shown in (13).

$$f_1 = \sum_{i=1}^{n} x_i^2 \tag{13}$$

- **The banana (Rosenbrock) Function**: the name of this function comes from its shape, which seems to be like a banana. The mathematical description of this function is shown in (14). The Rosenbrock function has a known value of global minimum at (1, 1) with an optimal value of zero.

$$f_2(x) = \sum_{i=1}^{n-1} [100 \cdot (x_{i+1} - x_i^2) + (x_i - 1) \tag{14}$$

- **The Rastrigin function**: this function has several local minima, which has a known value of global minimum at (0, 0) with an optimal value of zero. Equation (15) shows the mathematical description of this function.

$$f_3(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos 2\pi x_i + 10) \tag{15}$$

- **2^n Minima**: the shape and the final optimal result of this benchmark function are based on the size of search domain. The mathematical description of this function is summarized in the following equation:

$$f_{4(x)} = \sum_{i=1}^{n} (x_i^4 - 16 \cdot x_i^2 + 5 \cdot x_i) \tag{16}$$

- **EggCrate function**: is the function of multi-model minimization problem, which has several local minima. This function has a known value of global minimum at (0, 0) with an optimal value of zero. Equation (17) shows the mathematical description of this function.

$$f_{5(x)} = x_1^2 + x_2^2 + 25\left(\sin^2 x_1 + \sin^2 x_2\right) \tag{17}$$

- **Sum Squares Function**: this function has a global minimum, but not has a local minimum. Sometimes this function is called Axis Parallel Hyper-Ellipsoid (APHE) function. This function has a known value of global minimum at (0, 0) with an optimal value of zero. The mathematical description of this function summarizes in the following equation:

$$f_6(\mathrm{x}) = \sum_{i=1}^{d} \mathrm{ix}_\mathrm{i}^2 \tag{18}$$

Where d is the value of the problem dimensions.

For each function, three various dimensional sizes are tested; 30, 20, and 10. These dimensional sizes are considered as standard dimensional sizes for evaluation of the previously mentioned benchmark functions [49–52]. Moreover, three different generations of the same population size are tested with those dimensions. Each function has the greater possible maximum value denoted by X_{max} in which equals to V_{max}. X_{max} and V_{max} are listed in Table 1. The third column in Table 1 presents the search domain for each function, where the dimension of each function is represented by n.

Table 1. Functions X_{Max} and V_{Max} with search domain

Function	$X_{max} = V_{max}$	Search domain
Sphere *f1*	100	$[-100, 100]^n$
Rosenbrock *f2*	100	$[-15, 30]^n$
Rastrigin *f3*	10	$[-2.56, 5.12]^n$
2^n Minima *f4*	10	$[-5, 5]^n$
EGGCrate *f5*	10	$[-5, 5]^n$
Sum Squares Function *f6*	10	$[-10, 10]^n$

The test environment was Intel dual-core CPU, 2 GB Ram. The MatLab version 7.0.4 has been used to implement the proposed algorithm. The outcomes of the proposed SSO method have been compared with the results of four methods such as GA, PGA, PSO, and

APSO algorithm. We use standard parameters during the simulation and evaluation of benchmark functions, where the parameters of SSO, PSO, APSO, GA and PGA algorithms are summarized in Table 2. We use standard variables and parameters for PSO and GA algorithms, which the values of C_1 and C_2 in PSO are set to 2 as in [53], while the probability of both crossover and mutation in GA are set to 0.8 and 0.05, respectively as in [54]. The parameters for both PGA and APSO are set as in [15, 16].

Table 2. Parameters of SSO, PSO, APSO, GA and PGA

Parameters	Value
SSO	
D: is a velocity damping factor	In range (0, 1)
pH	In range (7, 14)
Temperature	In range (35.5, 38.5)
population sizes	20 and 40
Numbers of generations	100, 500, and 1000
Mutation probability	1/a where a is the size of variable
PSO	
C_1 and C_2	2
Random range	In range (0, 1)
Population sizes	20 and 40
Numbers of generations	100, 500, and 1000
APSO	
r	In range (0, 1)
β	In range (0.2, 0.7)
α	In range (0.1, 0.5)L
δ	In range (0.1.99)
Population sizes	20 and 40
Numbers of generations	100, 500, 1000
GA	
Mutation probability	0.05
Crossover probability	0.8
Population sizes	20 and 40
Numbers of generations	100, 500, 1000
PGA	
Mutation probability	0.05
Crossover probability	0.8
Population sizes	20 and 40
Numbers of generations	100, 500, 1000

Overall, 10 runs for every tested function is conducted. The experimental outcomes for SSO, GA, PGA, PSO, and APSO algorithms are presented in Table 3. In the table, AVG represents the average final best value, and f(x) is the best realizable fitness value for the proposed functions. We follow a standard strategy for comparing the proposed

Table 3. Experimental results where the optimal value for f1 to f6 is zero except for f4. The best value for f4 is −156.6647

Algorithm	Function	Population size	Generation	The dimension of the function (n)	The best achievable fitness value f(x)	Average final best value (Avg)
SSO	Sphere *f1*	20	100	10	0.00000	0.00000
			500	20	0.00000	0.00000
			1000	30	0.00000	0.00000
	Rosenbrock *f2*	40	100	10	0.00000	3.15100
			500	20	0.00000	10.386E-08
			1000	30	0.00000	3.326E-07
	Rastrigin *f3*	40	100	10	0.00471	0.504261
			500	20	0.00406	0.495542
			1000	30	0.00226	0.495918
	2^n Minima *f4*	40	100	10	−156.6647	−156.6647
			500	20	−156.6647	−156.6647
			1000	30	−156.6647	−156.6647
	EGGCrate *f5*	40	100	10	0.00000	0.01562
			500	20	0.00000	0.01465
			1000	30	0.00000	0.013442
	Sum Squares *f6*	40	100	10	0.00000	0.00000
			500	20	0.00000	0.00000
			1000	30	0.00000	0.00000
PSO	Sphere *f1*	20	100	10	0.00000	0.00000
			500	20	0.00000	0.00000
			1000	30	0.00000	0.00000
	Rosenbrock *f2*	40	100	10	0.00000	4.53143
			500	20	0.00000	3.92E-09

(*continued*)

Table 3. (*continued*)

Algorithm	Function	Population size	Generation	The dimension of the function (n)	The best achievable fitness value f(x)	Average final best value (Avg)
			1000	30	0.00000	5.62E-08
	Rastrigin *f3*	40	100	10	0.3864	5.65491
			500	20	0.00000	4.59949
			1000	30	0.9950	4.89716
	2^n Minima *f4*	40	100	10	−156.6647	−145.15242
			500	20	−156.6647	−156.6647
			1000	30	−156.6647	−156.6647
	EGGCrate *f5*	40	100	10	0.00000	5.3337
			500	20	0.00000	3
			1000	30	0.00000	5.0001
	Sum Squares *f6*	40	100	10	0.00000	0.00000
			500	20	0.00000	0.00000
			1000	30	0.00000	0.00000
APSO	Sphere *f1*	20	100	10	3.4095e-032	2.55E-30
			500	20	3.7039e-156	3.04E-151
			1000	30	2.1229e-312	4.81E-303
	Rosenbrock *f2*	40	100	10	0.007695	3.583723
			500	20	0.006202	2.202E-07
			1000	30	7.6039E-010	7.6039E-010
	Rastrigin *f3*	40	100	10	0.99496	0.99496
			500	20	0.99496	0.99496
			1000	30	0.99496	0.9950
		40	100	10	−156.6647	−156.6647

(*continued*)

Table 3. (*continued*)

Algorithm	Function	Population size	Generation	The dimension of the function (n)	The best achievable fitness value f(x)	Average final best value (Avg)
	2^n Minima *f4*		500	20	−156.6647	−156.6647
			1000	30	−156.6647	−156.6647
	EGGCrate *f5*	40	100	10	8.0653e-030	1.91E-28
			500	20	8.2744e-155	1.79E-149
			1000	30	1.2512e-309	4.6299e-305
	Sum Squares *f6*	40	100	10	1.0648e-031	2.71E-29
			500	20	3.1837e-155	6.99E-151
			1000	30	1.7222e-310	8.06E-303
GA	Sphere *f1*	20	100	10	0.0000	0.14706
			500	20	0.0000	0.01623
			1000	30	0.0000	0.02758
	Rosenbrock *f2*	40	100	10	0.0000	0.85000
			500	20	0.0000	5.98763
			1000	30	0.0000	7.0303
	Rastrigin *f3*	40	100	10	0.0005	0.01746
			500	20	0.0000	0.021285
			1000	30	0.0000	0.023166
	2^n Minima *f4*	40	100	10	0.0000	0.035010
			500	20	0.0000	0.038100
			1000	30	0.0000	0.0319867
	EGGCrate *f5*	40	100	10	0.0000	0.006848
			500	20	0.0000	0.006184
			1000	30	0.0000	0.010755

(*continued*)

Table 3. (*continued*)

Algorithm	Function	Population size	Generation	The dimension of the function (n)	The best achievable fitness value f(x)	Average final best value (Avg)
	Sum Squares *f6*	40	100	10	0.0000	0.011384
			500	20	0.0000	0.00773
			1000	30	0.0000	0.009923
PGA	Sphere *f1*	20	100	10	0.0000	1.004675E-4
			500	20	0.0000	1.000004E-4
			1000	30	0.0000	9.928486E-5
	Rosenbrock *f2*	40	100	10	0.0000	0.007695
			500	20	0.0000	5.11542
			1000	30	0.0000	5.98623
	Rastrigin *f3*	40	100	10	0.0000	0.0041466
			500	20	0.0000	0.00129499
			1000	30	0.0000	2.8672288E-4
	2^n Minima *f4*	40	100	10	−0.0127686	−0.000898955
			500	20	−3.1567839	−0.159241
			1000	30	−13.716409	−8.67130934
	EGGCrate *f5*	40	100	10	0.0000	0.008854915
			500	20	0.0000	0.004965030
			1000	30	0.0000	0.007119022
	Sum Squares *f6*	40	100	10	0.0000	5.0914775E-5
			500	20	0.0000	5.0001215E-5
			1000	30	0.0000	5.0000061E-5

algorithms, which the number generations and iterations have been changed through the evaluation. This strategy of comparison is used to evaluate many well-known methods such as GA [48], Artificial Bee Colony (ABC) algorithm [49], Hurricane Search Algorithm (HSA) [50], and PSO [52].

Depends on the results from Table 3, the algorithms can be ranked from best achievable fitness value to worse achievable fitness value of all benchmark functions as in Table 4.

Table 4. The algorithms ranking from the best achievable value to the worse achievable value of each benchmark function

Function	The algorithms ranking from the best achievable fitness value to the worse achievable fitness value
Sphere f1	SSO and PSO have the same rank (first rank), followed by APSO, PGA, and GA respectively
Rosenbrock f2	SSO, PSO, APSO, PGA, and GA respectively
Rastrigin f3	PGA, GA, SSO, APSO, and PSO respectively
2n Minima f4	SSO and APSO have the same rank (first rank), followed by PSO, PGA, and GA respectively
EGGCrate f5	APSO, PGA, GA, SSO, and PSO respectively
Sum Squares f6	SSO and PSO in the same rank (first rank), APSO, PGA, and GA respectively

At the end, we can compare between SSO, GA, PGA, PSO, and APSO in Table 5 [33, 39, 55].

Table 5. Comparison between SSO, GA, PGA, PSO, and APSO [33, 39, 55]

Criteria of comparison	GA and PGA	PSO and APSO	SSO
Type of procedure	Discrete approaches	Continuous approaches	Continuous approaches
Type of a metaphor	Darwinian's notion of evolution utilized to biology, which mimics the structure of chromosome and its evolution	Interaction of social groups, which mimics the mobility of swarm of birds while searching for food	Natural fertilization approach, which mimics the motility of sperm flock through the fertilization operation
Use crossover operation	Use different types of crossover operations such as Simulated Binary Crossover (SBX)	Do not use operations of crossover	Do not use operations of crossover
Influence of population size or swarm size on solution time	Exponential	Linear	Linear
Best solution affects population	Deal with each individual independently	Use the solution of swarm leader (best solution) to add it for other individual solutions	Use the value of winner (the best solution) as a reference value to guide other members in the flock to adjust their velocities
Average fitness value cannot get worse	Average fitness will not be worse because the individual will be ranked from the best to the worse. The best individuals will be reserved for next step while the worst will be eliminated	Average fitness will not be worse because the velocity of the leader of the swarm (best solution) will be added to all other velocities in the swarm	Average fitness will not be worse because all members in the swarm will use the velocity of a winner (optimal solution) as a reference value

5 Conclusion

In this paper, SSO method has been reviewed to optimize different kinds of standard optimization problems. To prove its efficiency, SSO is evaluated with different benchmark models with various dimensions. We compare between proposed algorithm and other algorithms in finding the average final best value of all benchmark functions at 100, 500 and 1000 generations. The results show the capability of the proposed algorithm in optimizing different nonlinear models, as it outperforms PSO in finding the average final best value of Rosenbrock of 100 generations, Rastrigin of all generations, 2^n Minima functions when the number of generations equal to 100, and

EGGCrate function of all generations, while it outperforms APSO in Rosenbrock, Rastrigin, and sum squares function of all generations. In addition, SSO outperforms GA, or PGA in finding the average final best value of 2^n Minima, and sum squares function on all generations. Moreover, SSO outperforms GA, and PGA in finding the average final best value of Rosenbrock function of 100 and 500 generations. In the future works, we will use the concept of the Memetic Algorithms (MAs) to hybridize between SSO and GA in order to enhance the method convergence.

Acknowledgments. The authors acknowledge University of Malaya for the financial support (University Malaya Research Grant (RP036A-15AET) and facilitating in carrying out the work.

Authors' Contribution. The work was deduced from Hisham's Ph.D. thesis as Dr. Mohd Yamani Idna Idris and Dr. Ismail Ahmedy supervised him along his study.

References

1. Li, X., Yin, M.: An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Adv. Eng. Softw. **55**, 10–31 (2013)
2. Lagaros, N.D., Plevris, V., Papadrakakis, M.: Neurocomputing strategies for solving reliability-robust design optimization problems. Eng. Comput. **27**, 819–840 (2010)
3. Duan, H., Zhao, W., Wang, G., Feng, X.: Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO. Math. Probl. Eng. **2012**, 1–22 (2012)
4. Gandomi, A.H., Yang, X.S., Talatahari, S., Alavi, A.H.: Metaheuristic applications in structures and infrastructures. Newnes (2013)
5. Blowers, M., Mendoza-Schrock, O.: Machine intelligence and bio-inspired computation: theory and applications VII. In: Proceedings of SPIE, pp. 1–8 (2013)
6. da Silva Maximiano, M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Multiobjective metaheuristics for frequency assignment problem in mobile networks with large-scale real-world instances. Eng. Comput. **29**, 144–172 (2012)
7. Kaveh, A., Talatahari, S.: Hybrid charged system search and particle swarm optimization for engineering design problems. Eng. Comput. **28**, 423–440 (2011)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley, New York, NY (1989)
9. Gandomi, A.H., Yang, X.-S., Talatahari, S., Deb, S.: Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization. Comput. Math Appl. **63**, 191–200 (2012)
10. Gandomi, A.H., Talatahari, S., Yang, X.S., Deb, S.: Design optimization of truss structures using cuckoo search algorithm. Struct. Des. Tall Spec. Buildings **22**, 1330–1349 (2013)
11. Gandomi, A.H., Alavi, A.H.: Multi-stage genetic programming: a new strategy to nonlinear system modeling. Inf. Sci. **181**, 5227–5239 (2011)
12. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**, 28–39 (2006)
13. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. **39**, 459–471 (2007)
14. Shi, Y.: Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 81–86 (2001)

15. Wang, G.G., Hossein Gandomi, A., Yang, X.S., Hossein Alavi, A.: A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. Eng. Comput. **31**, 1198–1220 (2014)
16. Muhlenbein, H.: Evolution in time and space-the parallel genetic algorithm. Found. Genet. Algorithms **1**, 1–22 (1991)
17. Shehadeh, H.A., Ahmedy, I., Idris, M.Y.I.: Sperm swarm optimization algorithm for optimizing wireless sensor network challenges. In: International Conference on Communications and Broadband Networking (ICCBN 2018), pp. 53–59. Singapore, 24–26 February 2018
18. El-Hamrawy, S., Fawzy, H.E.D., Al-Tobgy, M.: Optimum Design for Close Range Photogrammetry Network Using Particle Swarm Optimization Technique, vol. 13, pp. 17–23 (2016)
19. Holland, J.H.: Genetic algorithms. Sci. Am. **267**, 66–73 (1992)
20. Paulinas, M., Ušinskas, A.: A survey of genetic algorithms applications for image enhancement and segmentation. Inf. Technol. Control **36**, 278–284 (2007)
21. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. **3**, 95–99 (1988)
22. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. Urbana **51**, 61801 (1991)
23. Goldberg, D.E., Deb, K., Clark, J.H.: Accounting for noise in the sizing of populations. Whitley **2419**, 127–140 (2014)
24. Forrest, S., Mitchell, M.: What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation. Mach. Learn. **13**, 285–319 (1993)
25. Meetei, K.T.: A survey: swarm intelligence vs. genetic algorithm. Int. J. Sci. Res. **3**, 231–235 (2014)
26. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer Science & Business Media, Berlin (2013)
27. Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA (2001)
28. Panigrahi, B.K., Suganthan, P.N., Das, S., Dash, S.S.: Swarm, evolutionary, and memetic computing. In: 4th Springer International Conference, SEMCCO 2013, pp. 19–21. Chennai, India, December, 2013
29. Ab Aziz, N.A., Ibrahim, Z.: Asynchronous particle swarm optimization for swarm robotics. In: Procedia Engineering, vol. 41, pp. 951–957 (2012)
30. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: International Conference on Evolutionary Programming, pp. 591–600 (1998)
31. Gandomi, A.H., Yun, G.J., Yang, X.-S., Talatahari, S.: Chaos-enhanced accelerated particle swarm optimization. Commun. Nonlinear Sci. Numer. Simul. **18**, 327–340 (2013)
32. Hassan, R., Cohanim, B., De Weck, O., Venter, G.: A comparison of particle swarm optimization and the genetic algorithm. In: Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference, p. e21 (2005)
33. Kachitvichyanukul, V.: Comparison of three evolutionary algorithms: GA, PSO, and DE. Ind. Eng. Manage. Syst. **11**, 215–223 (2012)
34. Riko, S., Andreja, R.: Intelligent Control Techniques in Mechatronics-Genetic Algorithm, Retrieved on, vol. 3, pp. 20–32 (2013)
35. Bai, Q.: Analysis of particle swarm optimization algorithm. Comput. Inf. Sci. **3**, 180–184 (2010)
36. Hassanat, A.B., Al-Nawaiseh, N.A., Abbadi, M.A., Alkasassbeh, M., Alhasanat, M.B.: Enhancing genetic algorithms using multi mutations: experimental results on the travelling salesman problem. Int. J. Comput. Sci. Inf. Secur. **14**, 785–800 (2016)

37. Li, M., Du, W., Nian, F.: An adaptive particle swarm optimization algorithm based on directed weighted complex network. Math. Probl. Eng. **2014**, 1–7 (2014)
38. Rane, V.A.: Particle swarm optimization (PSO) algorithm: parameters effect and analysis. Int. J. Innovative Res. Dev. **2**, 8–16 (2013)
39. Shehadeh, H.A., Idris, M.Y.I., Ahmedy, I.: Multi-objective optimization algorithm based on sperm fertilization procedure (MOSFP). Symmetry **9**, 241 (2017)
40. Das Neves, J., Bahia, M.: Gels as vaginal drug delivery systems. Int. J. Pharm. **318**, 1–14 (2006)
41. Rodrigues, J.J., Caldeira, J., Vaidya, B.: A novel intra-body sensor for vaginal temperature monitoring. Sensors **9**, 2797–2808 (2009)
42. Borges, S.F., Silva, J.G., Teixeira, P.C.: Survival and biofilm formation of Listeria monocytogenes in simulated vaginal fluid: influence of pH and strain origin. FEMS Immunol. Med. Microbiol. **62**, 315–320 (2011)
43. Edmunds, M.W., Mayhew, M.S.: Pharmacology for the Primary Care Provider-E-Book. Elsevier Health Sciences (2013)
44. Christian Nicole. Available online: https://christiannicole72838.wordpress.com/2013/06/26/from-the-figure-for-women-to-understand-their-own-sexual-organs-gender/ (2013). Accessed on 21 Nov 2017
45. Health. Available online: http://health-of-people.blogspot.com/2011/03/interestingly-statistics-of-physical.html (2011). Accessed on 21 Nov 2017
46. Nalepa, J., Kawulok, M.: A memetic algorithm to select training data for support vector machines. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 573–580 (2014)
47. Krzeszowski, T., Wiktorowicz, K.: Evaluation of selected fuzzy particle swarm optimization algorithms. In: Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 571–575 (2016)
48. Sathya, S.S., Radhika, M.: Convergence of nomadic genetic algorithm on benchmark mathematical functions. Appl. Soft Comput. **13**, 2759–2766 (2013)
49. Banharnsakun, A., Achalakul, T., Sirinaovakul, B.: The best-so-far selection in artificial bee colony algorithm. Appl. Soft Comput. **11**, 2888–2901 (2011)
50. Rbouh, I., El Imrani, A.A.: Hurricane search algorithm a new model for function optimization. In: IEEE 5th International Conference on Information and Communication Systems (ICICS), pp. 1–5 (2014)
51. Gollapudi, S.V., Pattnaik, S.S., Bajpai, O., Devi, S., Bakwad, K.M.: Velocity modulated bacterial foraging optimization technique (VMBFO). Appl. Soft Comput. **11**, 154–165 (2011)
52. Shi, Y., Eberhart, R.C.: Empirical study of particle swarm optimization. In: IEEE Congress on Evolutionary Computation, CEC 99, pp. 1945–1950 (1999)
53. Zhang, W., Liu, Y.: Reactive power optimization based on PSO in a practical power system. In: Power Engineering Society General Meeting, 2004. IEEE, pp. 239–243 (2004)
54. Saravanan, R., Sachithanandam, M.: Genetic algorithm (GA) for multivariable surface grinding process optimisation using a multi-objective function model. Int. J. Adv. Manuf. Technol. **17**, 330–338 (2001)
55. Shehadeh, H.A., Idna Idris, M.Y., Ahmedy, I., Ramli, R., Noor, N.M.: The multi-objective optimization algorithm based on sperm fertilization procedure (MOSFP) method for solving wireless sensor networks optimization problems in smart grid applications. Energies **11**, 97 (2018)