# Self-adaptive Salp Swarm Algorithm for Optimization Problems

**Sofian Kassaymeh** ( samsaak@gmail.com )
Universiti Kebangsaan Malaysia https://orcid.org/0000-0003-0586-1961
**Salwani Abdullah**
Universiti Kebangsaan Malaysia
**Mohammed Al-Betar**
Al-Balqa' Applied University
**Mohammed Alweshah**
Al-Balqa' Applied University
**Mohamad Al-Laham**
Al-Balqa' Applied University
**Zalinda Othman**
Universiti Kebangsaan Malaysia

**Research Article**

**Keywords:** Salp Swarm Algorithm, Initial Population Diversity, Self-Adaptive Parameters Tuning, Swarm Algorithms, Optimization, Metaheuristic

**Posted Date:** May 2nd, 2022

**DOI:** https://doi.org/10.21203/rs.3.rs-1600365/v1

# Self-Adaptive Salp Swarm Algorithm for Optimization Problems

Sofian Kassaymeh [1a], Salwani Abdullah[a], Mohammed Azmi Al-Betar[b], Mohammed Alweshah[c,d], Mohamad Al-Laham[c], Zalinda Othman[a]

[a]*Data Mining and Optimization Research Group, Center for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, Bangi Selangor, Malaysia*
p86165@siswa.ukm.edu.my, salwani@ukm.edu.my, zalinda@ukm.edu.my
[b]*Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, Irbid, Jordan.*
mohbetar@bau.edu.jo
[c]*Department of Computer Science, Prince Abdullah bin Ghazi Faculty of Communication and Information Technology, Al-Balqa Applied University, Salt, Jordan.* weshah@bau.edu.jo, Dr_laham@bau.edu.jo
[d]*Artificial Intelligence Department,College of Information Technology, Aqaba University of Technology, Aqaba, Jordan.*

**Abstract**

In this paper, an enhanced version of the Salp Swarm Algorithm (SSA) for global optimization problems was developed. Two improvements have been proposed: (i) diversification of the SSA population referred as $\mathbf{SSA}_{std}$, (ii) SSA parameters are tuned using a self-adaptive technique based Genetic Algorithm (GA) referred as $\mathbf{SSA}_{GA-tuner}$. The novelty of developing a self-adaptive SSA is to enhance its performance through balancing search exploration and exploitation. The enhanced SSA versions are evaluated using twelve benchmark functions. The diversified population of $\mathbf{SSA}_{std}$ enhances convergence behavior, and self-adaptive parameter tuning of $\mathbf{SSA}_{GA-tuner}$ improves the convergence behavior as well, thus improving performance. The comparative evaluation against nine well-established methods shows the superiority of the proposed SSA versions. The enhancement amount in accuracy was between 2.97% and 99% among all versions of algorithm. In a nutshell, the proposed SSA version shows a powerful enhancement that can be applied to a wide range of optimization problems.

*Keywords:*
Salp Swarm Algorithm, Initial Population Diversity, Self-Adaptive Parameters Tuning, Swarm Algorithms, Optimization, Metaheuristic.

## 1. Introduction

Algorithms of type swarm intelligence are inspired from foraging behavior or biological evolution in nature simulation [1, 2, 3, 4]. The recent swarm-based intelligence algorithms are Whale Optimization Algorithm (WOA) [5, 6], Grasshopper Optimization Algorithm (GOA) [7, 8], Artificial Bee Colony (ABC) [9, 10, 11], Ant Colony Optimization (ACO) [12], Particle Swarm Optimization (PSO) [13], Biogeography-Based Algorithm [14, 15], Bacterial Foraging Optimization(BFO) [16, 17], Grey Wolf Optimizer (GWO) [18, 19, 20], Fruit Fly Optimization (FOA) [21, 22], Kidney-Inspired Algorithm (KA)[23, 24], Firefly Algorithm [25, 26], and Harris Hawk Optimizer (HHO) [27]. The common features among these algorithms are their collaborative behavior strategies. They are able to achieve the evolution principle through iterative process where the current swarm is improved by attracting them to the local or global best solutions found in previous generations. Therefore, swarm-based intelligence algorithms can intelligently strike the right balance between diversification and intensification aspects [28, 29].

Due to their success features, swarm-based intelligence algorithms have been widely tailored for various types of optimization problems. However, the efficiency of these algorithms is directly affected by the search space nature of optimization problems [30]. Therefore, the theoretical aspects of the solvent algorithm could

[1]Corresponding Author

be improved in line with the search space properties of such an optimization algorithm. The improvements are normally on either parameter settings or operator behavior. Sometimes, the improvement can be achieved by hybridizing such algorithms with other algorithms to enhance convergence characteristics. In another perspective, these algorithms do not have the same structure and searching mechanism. They are different in properties, and they also behave differently based on the problem under consideration. For instance, it is a remark that the BFO has complex structure while PSO has so simple structure [31], at the same time the PSO converges easy but ACO converges slowly [32]. Moreover, exploration and exploitation for ABC and BFO are well-organized and have better capability whereas it is poor in for ACO [33]. Next, ABC and ACO are strongly affected and sensitive to parameter setting and initial values, instead PSO sensitive to problem dimension. Finally, ABC has strong randomness behavior with low accurate outcomes [34, 35, 36].

Based on "No Free Lunch" (NFL) theorem [37], no algorithm has the ability to handle all types of problems. Several works were attempts to improve the performance of swarm intelligence algorithms to get efficient and accurate outcomes. A new swarm intelligence algorithm was introduced a few years ago by [38] called Salp Swarm Algorithm (SSA). This algorithm mimics the behavior of the Salp fish in the sea. In addition, SSA has been verified by different engineering applications and several benchmark problems [38]. So, it has been applied to a lot of variety of optimization problems. For instance in [39, 40, 41, 42], the SSA algorithm is employed for Feature Selection Problem, whereas in [43], SSA is utilized for Polymer Electrolyte Membrane (PEM) fuel cells to extract the optimal parameters. Another noticeable use for SSA is designing the Complementary Metal–Oxide–Semiconductor (CMOS) analog integrated circuit (IC) by [44]. Also, [45] employs the SSA algorithm in calibrating the power system stabilizer for a power system for multi-machine. Next, the SSA is then used by [46] to estimate the activities of a chemical substance. Furthermore, in [47], the study of Short-Term Load Forecasting utilizing the SSA classifier was conducted. Also, the problem of fish image segmentation utilizing SSA in [48]. Moreover, SSA employed for predicting parameter values for the curve of soil water retention which proposed in [49], where SSA is proposed for parameter optimization of a detection model used for photovoltaic cell techniques [50]. Furthermore, utilizing SSA for power load frequency control and for load frequency control of power systems was done in [51] and [52] respectively.

Despite the variety of applications for the SSA algorithm, but it still suffers from some limitations. Like finding the right balance between diversification and intensification, [53, 42, 54, 55, 56, 42, 54], bad convergence accuracy [57, 58, 59], slow convergence rate [57, 58, 60, 61, 62, 63, 64, 65, 54, 42, 66, 67, 68, 69, 70], Lack of randomization components, [60], problems in discrete domain [71], deficiency of optimization ability, [65], and exploration ability [54].

A few improvements attempts on SSA were proposed in the last years. [55] enhances the SSA structure through tuning control parameters. In addition, a binary SSA algorithm was introduced to tackle the Arctan transformation problem [56]. Also, a chaos-induced SSA is proposed in [72] where the variables of chaotic initialized through a chaotic sequence which employed to substitute random variables. Furthermore, a Chaos-Induced and Mutation-Driven Schemes based SSA, and greedy criteria are hybridized with the SSA algorithm to improve convergence [54, 42].

Therefore, as mentioned previously, from the limitations of the SSA algorithm and the modifications made on it by several researchers, we conclude that the SSA algorithm does not work in the required form in its standard form, and it needs to be modified and enhanced in order to come up with satisfactory and competitive results. This motivates us to try to modify the SSA algorithm that enhances its performance.

The main objective of this work is to propose a new improved Salp Swarm Algorithm (SSA) by means of incorporating two improvement strategies in the initial population and the parameter tuning. To achieve such objective, the following contributions are made:

1. In the first improvement strategy, the initial population is chosen based on the diversity measurement where several populations are generated and the one with the highest diversity value is selected which is called diversified SSA algorithm ($\mathbf{SSA}_{std}$).
2. In the second strategy, self-adaptive parameter control is utilized in SSA parameters using a genetic algorithm to find the optimal parameter for SSA which is called self-adaptive SSA ($\mathbf{SSA}_{GA-tuner}$).
3. For verification and validation purposes, the proposed self-adaptive SSA algorithm is compared against the standard SSA algorithm and also with state of the art algorithms using twelve standard benchmark

functions.

4. The results prove the high impact of the self-adaptive SSA on the final outcomes.

The rest of the paper is organized as follows: Section 2 scans the Related Works. The fundamental background for the standard SSA and GA are discussed in Section 3. The proposed diversified SSA algorithm and self-adaptive SSA algorithms are described in Section 4. Results and discussions are thoroughly discussed and analyzed in Section 5. Finally, the conclusion and possible future directions are illustrated in Section 6.

## 2. Related Works

In this section, the two main concepts of diversity and parameter control are overviewed. These related to the main contributions of the present paper. Initially the related work of population diversity is provided in Section 2.1 while the proper and relevant literature about control parameters are given in Section 2.2.

### *2.1. Population Diversity*

As conventionally known, the initial parameter affects the convergence behavior of any swarm-based or evolutionary-based metaheuristic algorithms. When the population-based algorithm initiated with a strong population with appreciated diversity, the problem search space can be entirely navigated with an effective scan. Recall, the optimization domain concurs that the search shall be a concern with diversity in the initial stage and it will be turned toward equilibrium state until the search is stagnated in which the intensification state is reached. In general, two population diversity strategies can be categorized: on-line (diversity preservation) and off-line (diversity preservation) [73, 74]. Off-line diversity strategy defined as the process of initialized a diverse population before metaheuristic is executed. Whereas, diversity preservation strategy can be defined as the process of monitoring and keeping the population with as much as possible diverse through the algorithm execution. Population diversity must be maintained even before or through algorithm execution because metaheuristics performance is sensitive to the initial population diversity [75, 74].

Several research studies on initial population diversity were proposed to investigate its impact on algorithm performance and the final solution quality. Some of those studies are summarised in Table 1.

Table 1: Research Studies on Initial Population Diversity

| Study | What Proposed to Achieve Population Diversity |
|---|---|
| [76, 77] | Avoiding excessive gathering in promising optimal areas |
| [78, 79] | Avoiding premature convergence by use the dissipation method |
| [80, 81, 82] | Utilized niching techniques to accelerate the convergence |
| [83, 84, 85] | Adaptively calibrate swarm number |
| [86, 87] | Mutation method based individual level |

All the mentioned methods focus on increasing the initial population diversity in order to increases the robustness of the proposed algorithm toward premature convergence [77, 88, 89], not trapped in local optima [90], and achieve balance among exploitation and exploration [91].

### *2.2. Parameter Control Strategies*

Normally, the parameter settings of any optimization algorithm can be classified into two types: parameter tuning and parameter control [92]. Parameter tuning is the process of choosing the right parameter before the search. These parameter values will remain unchanged during the search. Normally parameter tuning is carried out by either experienced users or by exhausted *ad-hoc* experimental parameters study. On the other hand, parameter control defined as the process of finding the optimal parameter settings for the optimization algorithm during the search to empower its search process thus improving the final outcomes [93]. The main purpose of parameter controls is twofold: (i) to build a parameter-less optimization algorithm that can be used by naive users as a black-box. (ii) to make use of the full utilization of the algorithm

efficiency by striking the right balance among wide-area exploration and local-nearby exploitation during the search.

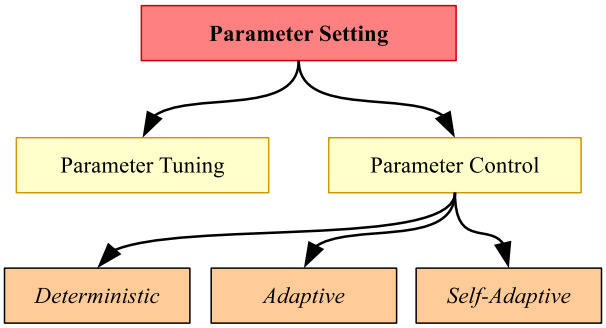


Figure 1: Global Taxonomy of Parameter Setting in EAs [94]

There are three types of parameter control strategies: deterministic, adaptive and self-adaptive illustrated in Figure 1. Deterministic parameter control modifies the value of the parameters during the search using normally the number of generations without feedback from the accumulative search process. While the adaptive parameter control is a strategy that updates the value of the parameters during the search based on feedback from the accumulative search. For example, when the search frequently improves the population, the parameters are updated to control their operator to focus on intensification rather than diversification. In contrast, when the search is stagnated and the population became ideal, the parameter values are updated to control their operator to focus on diversification rather than intensification. The third type is the self-adaptive or "*evolution of evolution*" parameter control in which the parameter values of the outer evolutionary algorithm is updated using another inner evolutionary algorithm. The inner evolutionary algorithm uses the set of parameters as a chromosome to be optimized and evaluated by the outer evolutionary algorithm. This type of parameter control strategy is the core contribution of this paper where the SSA is the outer evolutionary algorithm and GA is the inner evolutionary algorithm. Some researchers from the literature on parameter control strategies are summarized in Table-2.

Table 2: Research Studies on Parameter Control

| Strategy | Study | What Proposed to Achieve Parameter Control |
|---|---|---|
| Deterministic Parameter Control | [95] | particular amount of generations |
| | [96] | target computational time |
| | [97] | particular amount of fitness function evaluations |
| Adaptive Parameter Control | [98] | particular amount of generations |
| | [99] | target computational time |
| | [100] | particular amount of fitness function evaluations |
| Self-Adaptive Parameter Control | [101] | |
| | [102] | |
| | [103] | |

## 3. Research Background

In order to provide a self-exploratory paper, this section presents the standard Salp Swarm Algorithm (SSA) in Section 3.1 and standard Genetic Algorithm (GA) in Section 3.2.

### *3.1. Fundamentals to Salp Swarm Algorithm*

Salp Swarm Algorithm (SSA) was proposed by [38]. It inspired the sea salps swarming behavior. Salp is considered as a type of Salpidae family and it has a cylindrical shape as shown in Figure 2 (a). In order

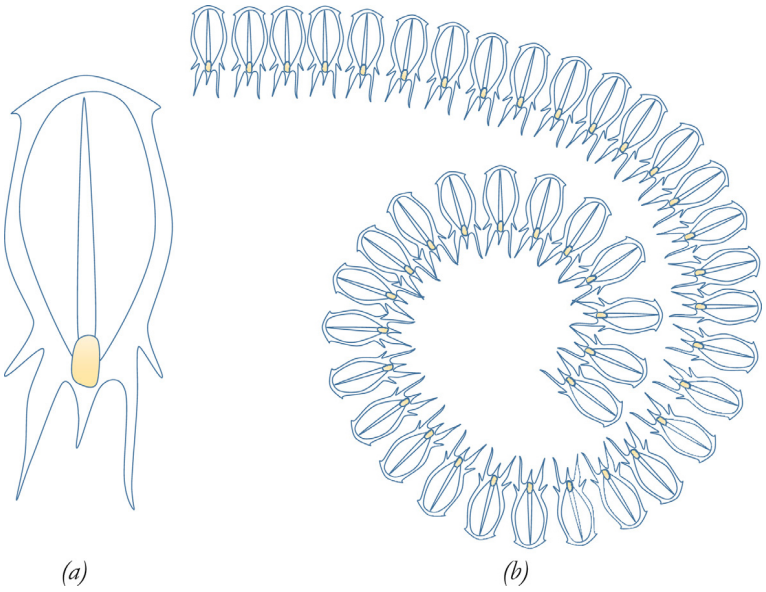


Figure 2: (a) Single salp, (b) Salps chain (salps swarm) [104]

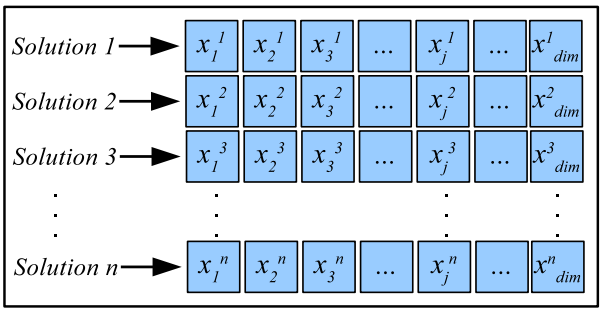


Figure 3: Standard SSA Algorithm Population Representation

to move in the sea, salps are able to form a chain, namely "*swarm chain*" shown also in Figure 2 (b). The swarm behavior assists salps for foraging and moving easily. The first salp in the chain is called the leader, and the rest of the salps are called the followers.

To elaborate, the leader has an important task to guide the swarm chain in movement and foraging. Therefore, the salps position is formulated as $dim - dimension$ in the search area, where $dim$ is the number of decision variables (or solution dimension) for a certain problem. Moreover, the salps position is saved in a matrix namely $x$. Furthermore, food source ($F$) in the search area is the main target of the salp chain.

SSA mechanism begins with a set of random positions for salps. Formally, the positions of the salps are generated using Eq.(5)

$$\mathbf{x}^i = rand(x_j^i) * (ub - lb) + lb \tag{1}$$

Where every solution is represented as $\mathbf{x}^i$ vector where $i \in (1, 2, \ldots, n)$ in which $n$ is the population size, $j \in (1, 2, \ldots, dim)$. Each decision variable $x_j^i \in [lb_j, ub_j]$ where $ub_j$ and $lb_j$ are the upper and lower bounds of decision variable $j$, respectively. A set of salps constitute one solution, and a set of solutions constitute one population as shown in Figure 3.

After computing the fitness value for every solution, the best solution can be found and appointed to source food ($F$). In addition, the leader movement is computed using Eq.(2).

$$\mathbf{x}_j^1 = \begin{cases} F_j + c_1((ub_j - lb_j)c_2 + lb_j) & c_3 \geq 0.5 \\ F_j - c_1((ub_j - lb_j)c_2 + lb_j) & c_3 < 0.5 \end{cases} \tag{2}$$

where $x_j^1$ represent the salp leader coordinates in the $j^{th}$ dimension, $F_j$ is the food source coordinates in the $j^{th}$ dimension, $ub_j$ announce the upper bound of $j^{th}$ dimension, $lb_j$ announce the lower bound of the $j^{th}$ dimension, $c_1$ computed using Eq.(3), $c_2$ and $c_3$ are random values.

It can be notable from Eq.(2) that the leader movement updated according to the food source. Also, the parameter $c_1$ which is computed using Eq.(3) is the most important parameter, thus it is responsible significantly for the balance among exploration and exploitation [40].

$$\mathbf{c}_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \tag{3}$$

where ($l$) and ($L$) are the current iteration and a maximum number of iterations respectively. The parameters $c_2$ and $c_3$ are random numbers uniformly initialized in the range of [0, 1]

On the other hand, follower salps positions are computed using Eq.(4).

$$\mathbf{x}_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right) \tag{4}$$

where $i \geq 2$ and $x_j^i$ represent the $i^{th}$ follower salp position in the $j^{th}$ dimension.

The simulation for salp swarm behavior is as follows. The algorithm starts the initialization stage by initializing a collection of salp chains representing a group of solutions, these solutions in combine considered as the initial population of the algorithm. In sequence, the fitness for all solutions is calculated, and the best solution is determined. After initialization, the SSA algorithm improvement stage starts by calculating the $c_1$, $c_2$, and $c_3$ parameters values. Then, the position for salps is updated, even by Eq.(2) for the first salp in the chain or by Eq.(4) for the rest of the salps in the chain. Updating solutions step is followed by checking whether the salps are still within the upper and lower limit range, if any salp is above the upper limits it is reset to the upper bound and if any salp below the lower limit it is reset to the lower bound. At this point, the fitness for updated solutions is calculated and compared with the fitness of the initial solutions and chose the best one as the best solution.

The above-mentioned steps are carried out iteratively until the termination criteria are met, of course except for the initializing step. At last, the Standard Salp Swarm Algorithm flowchart is presented in Figure 4, and the pseudo code is given in Algorithm 1.

### *3.2. Fundamentals to Genetic Algorithm*

Genetic Algorithm (GA) is a popular population-based evolutionary-based algorithm proposed by [105]. It is initiated with a population of individuals. Each individual has a set of genes. GA has conventionally utilized the survival of the fittest rule in the natural selection principle [106, 107]. Evolution after evolution, GA regenerates the current population using three main operators: selection, crossover, and mutation. Each GA gene is a decision variable and each individual is a solution, as shown in Figure 5. All individuals in the GA population have to be evaluated to get their fitness by utilizing what's called the objective function. For the purpose of promoting low fitting individuals, an elitism mechanism for selecting the best individuals are employed. Also, the probability of selecting poor solutions mechanism employed to raise the local optima prevention.

In addition, the GA algorithm considered as a reliable algorithm and trustworthy to find the global optimum [108, 109, 110], so that, its technique preserves the best solutions through all generation and utilize it to enhance the poor solutions. So, all the population individuals turn out to be better. Crossover among individuals leads to exploitation of the "*zone*" between the two parental solutions given. Also, mutation benefits the algorithm, where this operator modifies the genes inside the chromosomes randomly, which will preserve the population individual's diversity and raise the GA behavior of exploration. Furthermore,

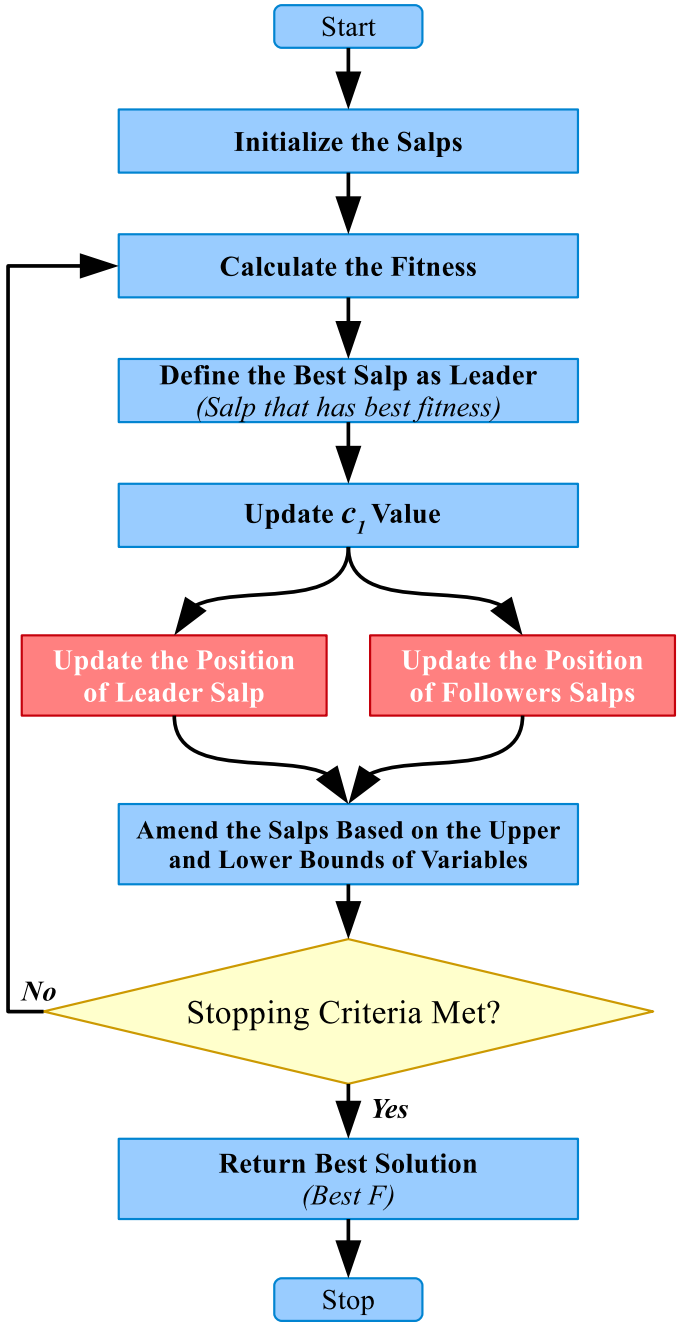


Figure 4: Standard SSA Algorithm Flow Chart

**Algorithm 1** Standard Salp Swarm Algorithm Pseudo Code

1: − − − − − − −**Stage 1: Initialization: Random Population** − − − − − − −
2: initialize a random initial population as $Pop_{init}$ using Eq.(5)
3: calculate the initial fitness of all solutions in $Pop_{init}$
4: find the best solution referred as $Sol_{best}$
5: − − − − −**Stage 2: Improvement: Salp Swarm Algorithm** − − − − −
6: set maximum number of iterations ($L$)
7: set counter $l \leftarrow 1$
8: **while** ($l < L$) **do**
9: update $c1$ using Eq.(3)
10: **for** each solution in $Pop_{init}$ **do**
11: update the first salp using Eq.(2)
12: update the remaining salps using Eq.(4)
13: **end for**
14: update $Pop_{init}$ referred as $Pop_{temp}$
15: **for** each salp in each solution in $Pop_{temp}$ **do**
16: **if** $x_j^i > ub$ **then**
17: $x_j^i = ub$
18: **else if** $x_j^i < lb$ **then**
19: $x_j^i = lb$
20: **end if**
21: **end for**
22: update $Pop_{temp}$ referred as $Pop_{new}$
23: calculate the fitness of all solutions in the updated population ($Pop_{new}$)
24: $l++$
25: **end while**
26: select best solution in $Pop_{new}$ referred as $Sol_{new}$
27: **if** $Sol_{new}$ is better than $Sol_{best}$ **then**
28: $Sol_{best} = Sol_{new}$
29: **end if**
30: return $Sol_{best}$

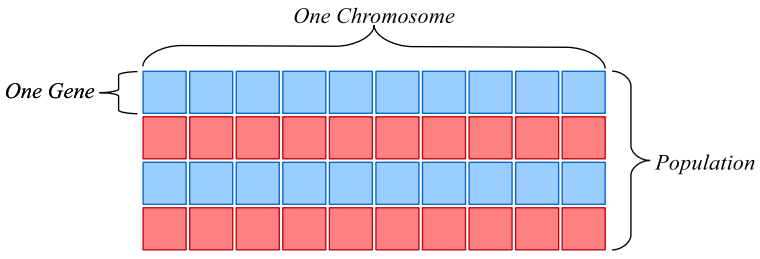


Figure 5: Standard GA Algorithm Population Representation

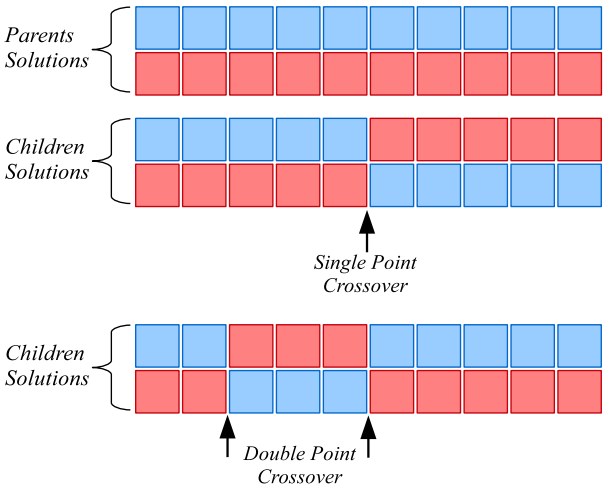


Figure 6: GA Algorithm Crossover Technique

the mutation operator may cause essential better solutions and guide other solutions to the global minima. Procedurally, GA has several steps to be executed, discussed as follows:

**Initial Population:** GA begins its process with a random population, which comprises multi individuals called chromosomes. Every chromosome has a group of variables that imitates the natural genes, as presented in Figure 5.

**Selection:** the main inspiration for the GA algorithm is natural selection. The fittest individual has the more chance to be selected for mating, which increases their genes contribution in the production of the next generation. The selection of individuals depends on their probability values, which in turn depends on the fitness values assigned by the GA algorithm.

**Crossover:** the crossover process is about an exchange of genes between two individuals (parent solutions) who have been pre-selected based on their fitness to generate two new individuals (children solutions), as seen in Figure 6. The two popular methods for crossover are single-point and double-point methods. This operator is normally controlled by crossover rate $\gamma_r$ where $\gamma_r \in [0, 1]$.

**Mutation:** The mutation is the process of altering single or multi genes in the children's solutions, which presented in Figure 7. Usually, the mutation rate set to be low because raising it may cause the GA algorithm to be just a random search technique. In addition to this, it takes advantage of the mutation that it preserves the diversity of the population by proposing more randomness and raising the possibility to prevent trapping in the local optimum. This operator is normally controlled by mutation rate $\mu_r$ where $\mu_r \in [0, 1]$.

In a nutshell, GA always begins its process with random individuals comprising its population, and across its process, it utilizes the early mentioned operators (Selection, Crossover, and Mutation) to enhance the population. Also, the best solution is considered the global optimum best approximation for the problem under solution. Finally, the high-level schemata of the Standard Genetic Algorithm are given in Algorithm 2.

## 4. Proposed Method

As aforementioned, in this paper, two main contributions are proposed to improve the performance of SSA:

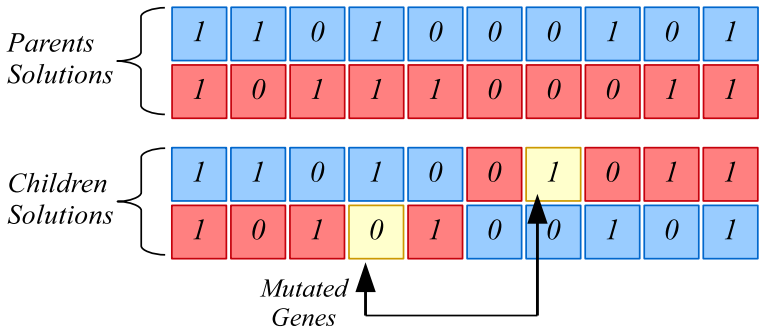


Figure 7: GA Algorithm Mutation Technique

**Algorithm 2** Standard GA Algorithm Pseudo Code

```
START
Initialize random population
Calculate fitness for all solutions in population
Compute fitness
for each solution in population do
    Selection
    Crossover
    Mutation
    Calculate fitness
    UNTIL population has converged
end for
STOP
Return best solution
```

1. **Initial Population Diversification of SSA**: This is achieved by generating multiple initial populations and chooses the most diversified one based on the statistical indications related to standard deviation. The diversified population selected is referred to as $\mathbf{SSA}_{std}$.
2. **Self-Adaptive SSA**: Incorporating the self-adaptive concepts in order to select the most appreciated parameters of SSA using the GA algorithm. The self-adaptive algorithm is referred to as $\mathbf{SSA}_{GA-tuner}$.

The following subsection will be thoroughly discussed the two contributions.

### *4.1. Diversification of Initial Population*

The standard SSA algorithm structure is improved by modifying the initial population initialization strategy. The modification includes a statistical indication based on the idea of computing the standard deviations of the initial populations. Therefore, the diversity of SSA is improved by means of striking the right balance between exploration and exploitation during the search. The proposed algorithm is referred to as $\mathbf{SSA}_{std}$.

Initially, multiple random populations as many as $(Max\#Pop)$ are generated using Eq.(5). This done in a loop of $k$ cycles as shown in Figure 8. At each $k$ cycle (say $i$), the standard deviation of the generated population $std(\mathbf{X}_k)$ is calculated. To elaborate, for every decision variable $(x_j^i)_{(k)}$ in the population $k$ the standard deviation is calculated using Eq.(6), and the average of standard deviations (i.e., $avg(std(x_j^i)), \forall i = (1, 2, \ldots, dim) \bigwedge \forall j = (1, 2, \ldots, n)$) of the decision variables are calculated using Eq.(7). Thereafter, a comparison between the average of standard deviations (i.e., $\mathbf{avg}(std(x_j^i)_{(k)})$) for all generated populations is conducted. The population with the highest average standard deviation is selected and used for SSA. The generation process of diversified initial population for SSA pseudo-coded found in Algorithm 3.

$$x^i = rand(x_j^i) * (ub - lb) + lb \tag{5}$$

$x^i$ represents the the $i^{th}$ generated solution, where $(i = 1, 2, \ldots, N)$ $(j = 1, 2, \ldots, dim)$. In addition, $N$ represents the size of the population. Also, the $dim$ represents the dimensions of the solution (size). The $ub$ and $lb$ represent the upper and lower bounds of the solution space.

$$\mathbf{std}(x^i) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x^i - \overline{x})^2} \tag{6}$$

$x^i$ represents the the $i^{th}$ generated solution. In addition, $\overline{x}$ represents the average of the generated solution. Also, the $n$ represents the population size.

$$\mathbf{avg}(std(x^i)_j) = \frac{\sum_{j=1}^{dim} std(x^i)}{dim} \tag{7}$$

where $std(x^i)$ is the standard deviation of the $i^{th}$ decision variable $x^i$, $(j = 1, 2, \ldots, dim)$, and $dim$ is the number of decision variables (or solution dimension).

### *4.2. Self-Adaptive Salp Swarm Algorithm*

The second contribution of this work implies proposing a self-adaptive salp swarm algorithm where the genetic algorithm (GA) is used in each iteration of the salp swarm algorithm to tune its parameters. The proposed algorithm is referred to as $\mathbf{SSA}_{GA-tuner}$. In $\mathbf{SSA}_{GA-tuner}$, GA plays a crucial rule in determining the optimal parameters for SSA.

The proposed $\mathbf{SSA}_{GA-tuner}$ has five steps as shown in the Flowchart visualized in Figure 10 and pseudo-coded in Algorithm 5. These steps can be thoroughly discussed as follows:

**Initialization** Initially the individual of GA is represented as a vector (i.e., $\boldsymbol{y} = (p_1, p_2)$) of length $d = 2$. The decision variables in the individual are the $p_1$ and $p_2$ which is the first and second parameters to determine $c_1$ presented in Eq.(9). To evaluate each individual, the SSA is used as a standard

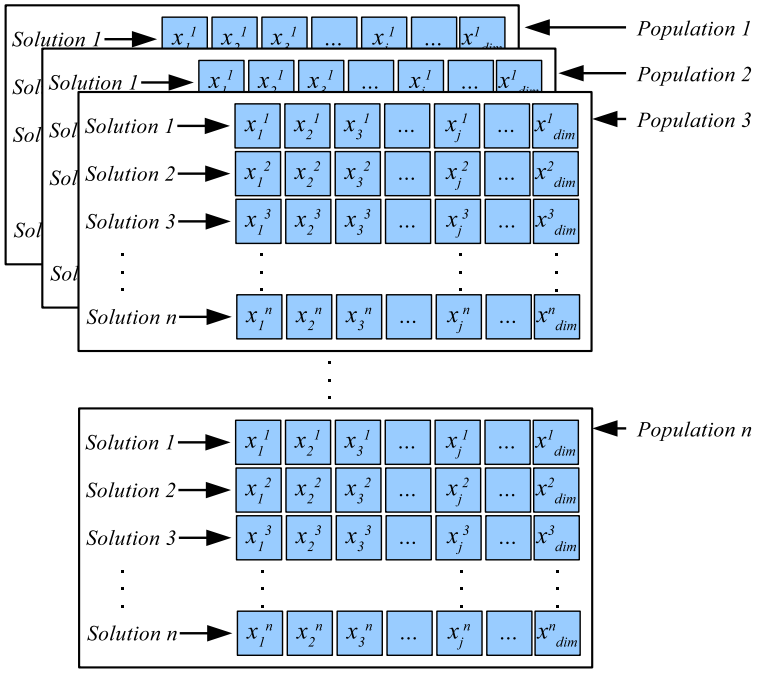


Figure 8: Multiple Random Populations Structure

**Algorithm 3** Generating a Diversified Initial Population

1: − − −**Stage 1: Initialization: Constructive Heuristic** − −−
2: set maximum number of initial population as $Max\#Pop$
3: set counter $k \leftarrow 1$
4: **while** $(k \leq Max\#Pop)$ **do**
5: calculate standard deviation for every decision variable $(x_j^i)_{(k)}$ in $Pop_{init_k}$ referred as $\mathbf{std}(x_j^i)_{(k)}$ using Eq.(6)
6: calculate the average of standard deviations $\mathbf{avg}(std(x_j^i)_{(k)})$ of the decision variables using Eq.(7)
7: $k++$
8: **end while**
9: select the best population based on highest value of $\mathbf{avg}(std(x_j^i)_{(k)})$ referred as $Pop_{init_{best}}$
10: calculate the fitness of all solutions in $Pop_{init_{best}}$
11: find the best solution referred as $Sol_{best}$
12: − − − − −**Stage 2: Improvement: Salp Swarm Algorithm** − − − −−
13: as in Algorithm 1 *(from line-6 to line-30)*

benchmark function with predefined population size and specific maximum number of iteration (i.e. $maxItr$). The results obtained by SSA for each individual is considered as the fitness function value. For GA, the initial population of size $(GA_{PopSize})$ is randomly generated with the discrete range of $p_1, p_2 \in (0, 1, \ldots, 15)$. This value range is selected after intensive experiments whereby this value range yields the best results.

The idea of the evolution of evolution can be used to implement the self-adaptation of parameters. Here the parameters to be adapted are encoded into the chromosomes and undergo mutation and recombination. The better values of these encoded parameters lead to better individuals, which in turn are more likely to survive and produce offspring and hence propagate these better parameter values.

**Selection** The proportional selection scheme (i.e., roulette wheel selection ) that is utilized the survival-of-

the-fittest principle is used to select the fittest individuals. In the proportional selection scheme fitness function of each individual is calculated using SSA. This is done by using any individual as an input parameter for SSA and the fittest solution produced is considered as the objective function value for that solution. The selection probability of each individual is calculated by the fitness function value of that individual relative to the fitness function values of the other individuals in the GA population ($GA_{Pop}$). Formally, let the $\varphi_i$ is the selection probability of the individual $i$. The value of $\varphi_i$ is calculated as in Eq. (8).

$$\varphi_i = \frac{f(\boldsymbol{x}_i)}{\sum_{j=1}^{PopSize} f(\boldsymbol{x}_j)} \tag{8}$$

Note that the $\sum_{j=1}^{PopSize} \varphi_i$ is unity. For example, Figure 9 pie-charts the probability of five individual $GA_{Pop} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5)$ where the fitness vector is $(f(\boldsymbol{x}_1) = 0.54, f(\boldsymbol{x}_2) = 1.5, f(\boldsymbol{x}_3) = 2.66, f(\boldsymbol{x}_4) = 3.22, f(\boldsymbol{x}_5) = 4.13)$. The selection probability of the each individual is represented as a portion in the pie-chart. In a nutshell, the larger portion means higher chance of selecting that individual.

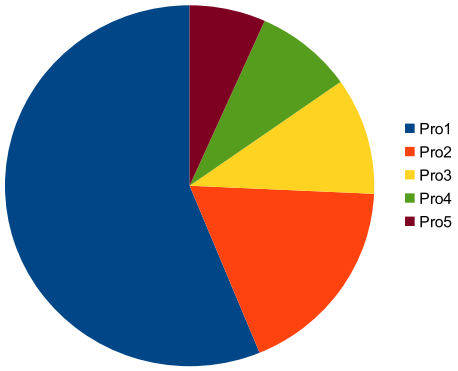


Figure 9: Pie-chart of proportional selection scheme example.

**Encoding** In the encoding step, the whole decision variables in the individuals stored in $GA_{Pop}$ are reformulated using a binary format. For example, let the individual be $\boldsymbol{x} = \{6, 14\}$, it will be reformulated to binary as follows: $\boldsymbol{x} = \{0110, 1110\}$. It is worth noting that, the individual is called a chromosome, and each bit of the chromosome is called a gene. Finally, Figure 5 illustrates the chromosome and gene structure.

**Crossover** The selected individuals pass to a crossover operator in which two encoded parents are randomly chosen. Thereafter, single and double point crossover is used as shown in Figure 6. In the single-point crossover, the parent solutions chromosomes exchanging after a randomly selected cut point to yield two new chromosomes. Where in the double-point crossover, two parents are chosen randomly. Therefore, two cut-points are pinned. The genes between the two cut-points are exchanged to yield two new chromosomes. The crossover rate $\gamma_r$ where $\gamma_r \in [0, 1]$ is used to determine the probability of using a crossover operator. The higher value of $\gamma_r$ closed to one leads to the use of a crossover operator for almost the entire population of individuals. This means that the genes will be heavily inherited between individuals. In the proposed method, $\gamma_r = 70\%$.

**Mutation** mutation is the next GA operator where one or more genes, based on mutation rate $m_r$ is altered in the chromosome to avoid similarity between solutions and to keep solutions away from local solutions. In addition, the mutation rate was assigned very low to ensure that the GA algorithm search process in not primitive random. An example of this operator is shown in Figure 7. From the figure,

it is clear that only a trivial change has occurred in the chromosome genes after the mutation process. Conventionally, $\mu_r$ is assigned by small value to control the search better. In the proposed method, $\mu_r = 1\%$.

**Decoding** The new chromosomes is decoded from binary into decimal format.

**Evaluation using SSA** To evaluate each individual, SSA is used. The gene values in each individual are used by SSA as initialized values for $p_1$ and $p_2$ using one benchmark function for all GA population. Thereafter, the optimal value obtained by SSA is the fitness function value for each solution. Note that to evaluate any individual, the SSA is repeated 31 replications and the average of the best-solutions obtained by all replications is calculated to be the fitness value. The pseudo-code for calculating fitness is given in Algorithm 4. It is worth mentioning that the diversified Initial Population strategy presented in Sec. 4.1 is used in SSA to generate the initial population.

**Algorithm 4** Evaluating GA Individuals $(p_1, p_2)$ using SSA Pseudo Code

set GA population size referred as $GA_{PopSize}$
set No of SSA Runs referred as $repetition$
set counter $j \leftarrow 1$
**while** $(j \leq GA_{PopSize})$ **do**
  set counter $k \leftarrow 1$
  **while** $(k \leq repetition)$ **do**
    Calculate $(SSA(f_j, Sol_k))$ using $Pop_{init_{best}}$
    Calculate $(best - average_{(f_j, Sol_k)})$
    $k++$
  **end while**
  set counter $i \leftarrow 1$
  **while** $(i \leq GA_{PopSize})$ **do**
    $sum(Soli) = sum(Sol_i) + best - average_{(f_j - Sol_i)}$
    $f(Sol_i) = sum(Sol_i)/repetition$
    $i++$
  **end while**
  $j++$
**end while**

**GA Termination Criteria** The selection, encoding, crossover, mutation, decoding and evaluation with elitism operators are repeated until the maximum number of generations ($GA_{MaxGen}$) is reached. After $GA_{MaxGen}$ is met, the best individual is selected to be the optimized parameter for SSA.

$$c_1 = p_1 e^{-\left(\frac{p_2 l}{L}\right)^2} \tag{9}$$

## 5. Results and Discussion

To evaluate the performance of the proposed algorithms, two sets of experiments are conducted. In the first set of experiments, the effect of the proposed diversified population in $\mathbf{SSA}_{std}$ is studied by comparing it against the standard SSA algorithm over twelve benchmark functions. In the second set of experiments, the effect of the self-adaptive tuning parameter in the diversified population $\mathbf{SSA}_{GA-tuner}$ is studied and compared against a diversified population $\mathbf{SSA}_{std}$ without self-adaptive tuning parameter, as well as the standard SSA algorithm using the same twelve benchmark functions. In order to comparatively evaluate the proposed method, nine comparative algorithms are used using ten benchmark functions. Finally, statistical evaluation is also conducted where the Wilcoxon Mann-Whitney Statistical test is used to provide statistical indications for significant results.

**Algorithm 5** **SSA**$_{GA-tuner}$ Pseudo Code

1: – – – – – – – – – **Stage 1: Initialization: Constructive Heuristic** – – – – – – – – –
2: as in Algorithm 3 *(from line-2 to line-11)*
3: – – – – – **Stage 2: Parameters Initialization: Using Genetic Algorithm** – – – – –
4: set GA maximum generation $GA_{MaxGen}$
5: set GA population size $GA_{PopSize}$
6: set counter $i \leftarrow 1$
7: **while** $(i \leq GA_{PopSize})$ **do**
8: generate random values for $(p_1, p_2)$ to form a GA population $GA_{Pop}$
9: calculate the fitness for $Sol_{best}$ using SSA
10: $i++$
11: **end while**
12: set counter $j \leftarrow 1$
13: **while** $(j \leq GA_{MaxGen})$ **do**
14: **Selection:** select 2-pairs of $(p_1, p_2)$ at random referred as (parents)
15: **Encoding:** encode the selected parents in binary format
16: **Crossover:** apply single or double point crossover {*considering probability*}
17: **Mutation:** mutate one or two digit randomly {*considering probability*}
18: **Decoding:** decode the generated offspring
19: calculate the fitness for each offspring using SSA
20: **if** offspring fitness better than parents fitness **then**
21: replace parent with offspring
22: **end if**
23: $j++$
24: **end while**
25: return best $(p_1, p_2)$
26: – – – – – – – **Stage 3: Improvement: Salp Swarm Algorithm** – – – – – – –
27: as in Algorithm 1 *(from line-6 to line-8)*
28: update $c1$ based on the best $(p_1, p_2)$ using Eq.(9)
29: as in Algorithm 1 *(from line-10 to line-30)*

### *5.1. Benchmark Functions*

The benchmark functions are grouped into two types, a uni-modal and multi-modal, and are listed with their mathematical formulations, boundaries, global optima, and dimension in the Table 4 and Table 5. In general, the uni-modal functions are convenient for examining the algorithm exploitation capabilities, where the multi-modal problems that have multi-local minima are more convenient for examining the algorithm exploration capability.

For the purpose of evaluating the performance of the proposed algorithms, a collection of parameter settings is given as shown in Table 3 as suggested in [38], and a collection of evaluation criteria was conducted in this work as follows:

- **Mean Value:** It is the average of best-obtained values over multiple experimental runs.
- **Standard Deviation (STD):** Show if the proposed algorithm has the ability to generate the best value for multiple experimental runs.

### *5.2. Effect of Diversified Population on* $\boldsymbol{SSA}_{std}$

The comparison among proposed diversified $\mathbf{SSA}_{std}$ and the standard SSA algorithm performance over 31 experimental runs are illustrated in Table 6. The performance measures of the obtained results are calculated, including mean and standard deviation for each benchmark function. It is notable that the $\mathbf{SSA}_{std}$ algorithm is able to obtain the best results and outperforms the standard SSA in almost all tested functions. Results of $\mathbf{SSA}_{std}$ algorithm demonstrate that the more diverse initial population has a remarkable positive impact on the quality of the algorithm's final results.

Table 3: Parameters Settings

| **Parameter** | **Standard SSA** | **SSA**$_{std}$ | **SSA**$_{GA-tuner}$ |
|---|---|---|---|
| SSA Max Iteration ($L$) | 500 | 500 | 500 |
| SSA Termination Criteria | $L$ | $L$ | $L$ |
| SSA Population Size ($N$) | 20 | 20 | 20 |
| SSA Decision Variables No. | based on - benchmark function | based on - benchmark function | based on - benchmark function |
| No. of Runs | 31 | 31 | 31 |
| Max No. of Initial Population ($Max\#Pop$) | – | 2000 | 2000 |
| GA Maximum Generation ($GA_{MaxGen}$) | – | – | 2000 |
| GA Termination Criteria | – | – | $GA_{MaxGen}$ |
| GA Population Size ($GA_{PopSize}$) | – | – | 31 |
| $p_1$ and $p_2$ | – | – | determined by GA |
| $c_1$ | calculated by Eq.(3) | calculated by Eq.(3) | calculated by Eq.(9) |
| $c_2$ and $c_3$ | random | random | random |
| ($lb$) and ($ub$) | based on - benchmark function | based on - benchmark function | based on - benchmark function |

Table 4: Benchmark Functions Configurations

| No. | Function | Type | Range | $f_{min}$ | Dimension |
|---|---|---|---|---|---|
| F1 | Sphere | unimodal | $[-100, 100]$ | 0 | 30 |
| F2 | Rastrigin | multimodal | $[-5.12, 5.12]$ | 0 | 10 |
| F3 | Ackley | multimodal | $[-32.768, 32.768]$ | 0 | 10 |
| F4 | Griewank | multimodal | $[-600, 600]$ | 0 | 10 |
| F5 | Rosenbrock | multimodal | $[-5, 10]$ | 0 | 10 |
| F6 | Bukin No.6 | multimodal | $[-15, -5]$ | 0 | 10 |
| F7 | Bohachevsky No.1 | unimodal | $[-100, 100]$ | 0 | 10 |
| F8 | Zakharov | unimodal | $[-5, 10]$ | 0 | 10 |
| F9 | Booth | unimodal | $[-10, 10]$ | -959.640 | 10 |
| F10 | Michalewicz | multimodal | $[0, pi]$ | -9.66015 | 10 |
| F11 | Eggholder | multimodal | $[-512, 512]$ | 0 | 10 |
| F12 | Himmelblau | multimodal | $[-6, 6]$ | 0 | 10 |

Table 5: Mathematical Definition of the Employed Benchmark Functions

| **No.** | **Mathematical Definition** |
|---|---|
| F1 | $F_1(x) = \sum_{i=1}^{n} x_i^2$ |
| F2 | $f(x, y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$ |
| F3 | $f(x) = -a.exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)) + a + exp(1)$ |
| F4 | $f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$ |
| F5 | $f(x, y) = \sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$ |
| F6 | $f(x, y) = 100\sqrt{\lvert y - 0.01x^2\rvert} + 0.01\lvert x + 10\rvert$ |
| F7 | $F(x, y) = x^2 + 2y^2 - 0.3cos(3\pi x) - 0.4cos(4\pi y) + 0.7$ |
| F8 | $f(x) = f(x_1, ..., x_n) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$ |
| F9 | $f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$ |
| F10 | $f(x) = -\sum_{i=1}^{d} sin(x_i)sin^{2m}(\frac{ix_i^2}{pi})$ |
| F11 | $f(x) = -(x[, 2] + 47) * sin(sqrt(abs(x[, 2] + (x[, 1]/2) + 47))) -$ $x[, 1] * sin(sqrt(abs(x[, 1] - (x[, 2] + 47))))$ |
| F12 | $f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$ |

Table 6: **Mean (avg)** and **Standard Deviation (std)** of best obtained results between standard SSA, $\mathbf{SSA}_{std}$, and $\mathbf{SSA}_{GA-tuner}$ algorithms

| No. | Function | Statistical Measure | Standard SSA | $SSA_{std}$ | $SSA_{GA-tuner}$ |
|---|---|---|---|---|---|
| F1 | Sphere | avg | 2.59E-04 | 1.92E-08 | **1.91E-31** |
| | | std | 4.61E-04 | 5.51E-09 | **1.05E-30** |
| F2 | Rastrigin | avg | 2.45E+01 | 1.92E+01 | **1.91E+01** |
| | | std | 8.66E+00 | 7.94E+00 | **6.78E+00** |
| F3 | Ackley | avg | 1.10E+00 | 5.56E-01 | **3.32E-01** |
| | | std | 1.07E+00 | 8.62E-01 | **6.94E-01** |
| F4 | Griewank | avg | 1.95E-01 | 2.06E-01 | **2.64E-02** |
| | | std | 1.56E-01 | 1.02E-01 | **1.03E-02** |
| F5 | Rosenbrock | avg | 5.66E+01 | 4.05E+01 | **1.04E+01** |
| | | std | 8.12E+01 | 7.77E+01 | **1.66E+01** |
| F6 | Bukin | avg | 8.06E-02 | 6.29E-02 | **8.00E-04** |
| | | std | 4.64E-02 | 3.93E-02 | **1.44E-02** |
| F7 | Bohachevsky | avg | 2.87E-11 | 9.52E-12 | **0.00E+00** |
| | | std | 3.59E-11 | 9.95E-12 | **0.00E+00** |
| F8 | Zakharov | avg | 1.80E-10 | 1.59E-11 | **1.92E-62** |
| | | std | 2.81E-10 | 8.45E-12 | **9.87E-62** |
| F9 | Booth | avg | 7.38E-14 | 2.17E-14 | **0.00E+00** |
| | | std | 8.35E-14 | 3.84E-14 | **0.00E+00** |
| F10 | Michalewicz | avg | -7.06E+00 | -7.27E+00 | **-7.55E+00** |
| | | std | 8.94E-01 | 8.00E-01 | **8.30E-02** |
| F11 | Eggholder | avg | -9.47E+02 | -9.51E+02 | **-9.60E+02** |
| | | std | 3.63E+01 | 2.25E+01 | **5.78E-13** |
| F12 | Himmelblau | avg | 2.36E-13 | 7.26E-14 | **1.58E-31** |
| | | std | 4.11E-13 | 1.01E-13 | **3.21E-31** |

* *best results in bold*

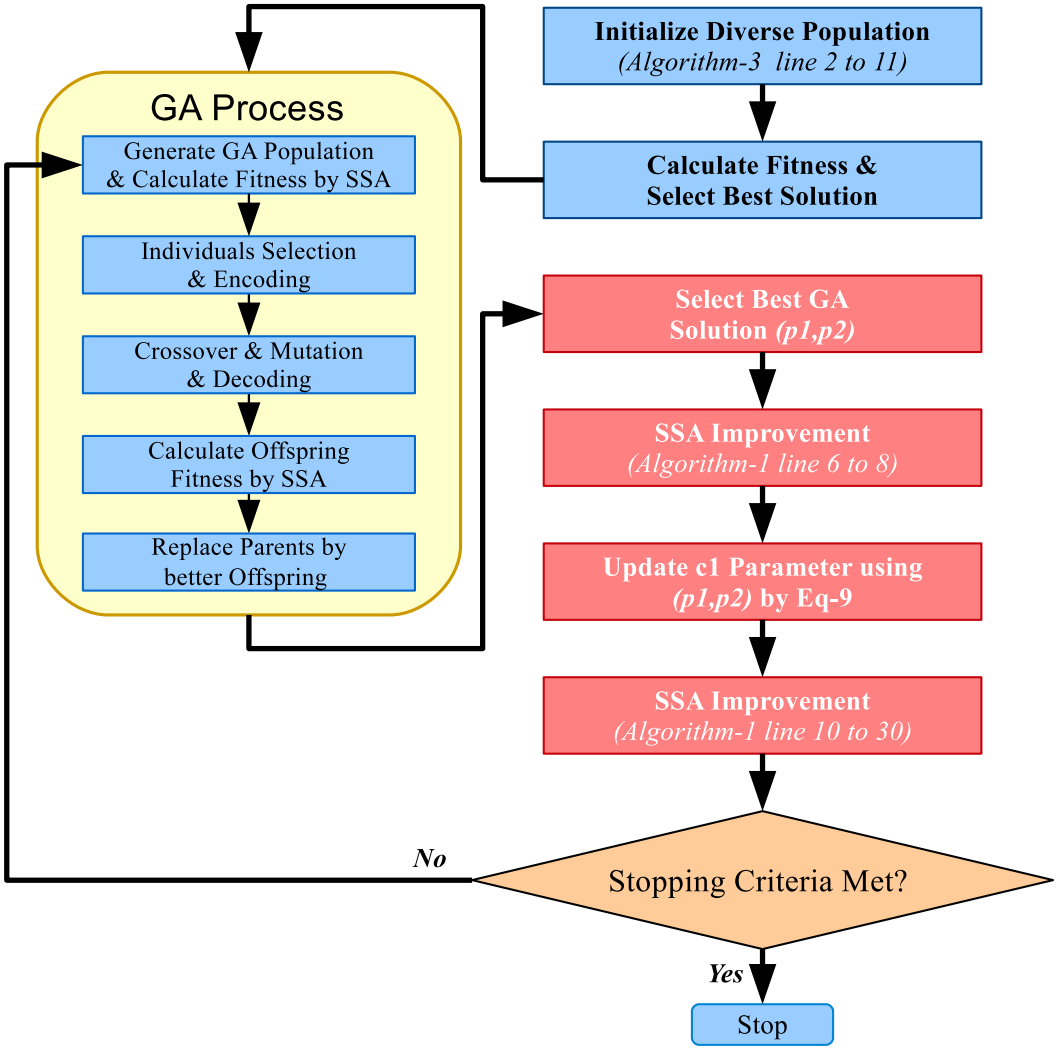


Figure 10: Proposed Self-Adaptive Salp Swarm Algorithm Flowchart

### *5.3. The Effect of Self-Adaptive Parameter Tuning on* $\boldsymbol{SSA}_{GA-tuner}$

The comparison between proposed tuned $\mathbf{SSA}_{GA-tuner}$ algorithm, diversified $\mathbf{SSA}_{std}$, and the standard SSA algorithm performance over 31 experimental runs are shown in Table 6. The performance measures of the algorithms are calculated, including mean and standard deviation for twelve benchmark problems. It is notable that the $\mathbf{SSA}_{GA-tuner}$ outperforms both $\mathbf{SSA}_{std}$ and the standard SSA for almost all tested functions. In addition, $\mathbf{SSA}_{GA-tuner}$ is able to obtain the best result in all functions in comparison with the other two algorithms. Results of $\mathbf{SSA}_{GA-tuner}$ proof that parameter tuning gives the algorithm the ability to deal with different population nature without proper experience from the users. Furthermore, parameter tuning enhances algorithm outcomes.

In order to validate the significance of the obtained results, the Wilcoxon Mann-Whitney Statistical test is conducted and its results are recorded in Table 7. These results are according to the best-obtained results. The statistical indications proof that the obtained results for $\mathbf{SSA}_{std}$ algorithm has significant difference (p-value $< 0.05$) in comparison with Standard SSA algorithm except on functions: F2, F4, F6, F9, and F10 and comparing with $\mathbf{SSA}_{GA-tuner}$ (except on functions: F4, F6, F7, F8, F9, F11, and F12). On the other

hand, obtained results for $\mathbf{SSA}_{GA-tuner}$ algorithm has significant differences comparing with Standard SSA algorithm (except for functions: F3, F4, F6, and F9). In addition, there is no significant difference among $\mathbf{SSA}_{std}$ and $\mathbf{SSA}_{GA-tuner}$ results, as there are seven functions with p-values greater than 0.05.

Table 7: p-values of Wilcoxon test for Standard SSA, $\mathbf{SSA}_{std}$ and $\mathbf{SSA}_{GA-tuner}$ Algorithms: best obtained results for employed benchmark functions

| No. | Function | $\mathbf{SSA}_{std}$ vs. Standard SSA | $\mathbf{SSA}_{GA-tuner}$ vs. Standard SSA | $\mathbf{SSA}_{GA-tuner}$ vs. $\mathbf{SSA}_{std}$ |
|---|---|---|---|---|
| F1 | Sphere | **0.000001** | **0.001197** | **0.000001** |
| F2 | Rastrigin | 0.377861 | **0.000001** | **0.000001** |
| F3 | Ackley | **0.001660** | 0.491077 | **0.021711** |
| F4 | Griewank | 0.121592 | 0.176324 | 0.680686 |
| F5 | Rosenbrock | **0.021077** | **0.000001** | **0.000002** |
| F6 | Bukin | 0.624195 | 0.543524 | 0.147018 |
| F7 | Bohachevsky | **0.000001** | **0.000001** | 1.000000 |
| F8 | Zakharov | **0.000001** | **0.000001** | 1.000000 |
| F9 | Booth | 0.124834 | 0.627534 | 1.000000 |
| F10 | Michalewicz | 0.066394 | **0.000001** | **0.000001** |
| F11 | Eggholder | **0.002470** | **0.033703** | 0.256839 |
| F12 | Himmelblau | **0.008151** | **0.008151** | 1.000000 |

** best results in bold*

### *5.4. Computational Time*

It is clear from Table-8 that the improved versions of the algorithm have more computational time than the standard algorithm. The increase in computational time is due to the algorithm performing additional tasks before proceeding with the main optimization process. For example, the $\text{SSA}_{std}$ algorithm, search for high diverse population before moving to the optimization process, and the $\text{SSA}_{GA-tuner}$ algorithm has two additional tasks in addition to the main optimization process (e.i., searching for a high diverse population and finding the optimal parameter values for a specific problem). In addition, it is noticeable that the $\text{SSA}_{GA-tuner}$ algorithm has a large increase in computational time over the rest of the algorithms, as this is due to the large time required to find parameter values.

Table 8: Computational Time

| No. | Function | Standard SSA | $\text{SSA}_{std}$ | $\text{SSA}_{GA-tuner}$ |
|---|---|---|---|---|
| F1 | Sphere | 0.6531415 | 1.0665227 | 39.9758610 |
| F2 | Rastrigin | 0.6529071 | 1.0726981 | 42.5873026 |
| F3 | Ackley | 0.6634043 | 1.0226595 | 43.7113842 |
| F4 | Griewank | 0.6342823 | 1.0413952 | 41.8555096 |
| F5 | Rosenbrock | 0.6050359 | 1.0282158 | 40.0824403 |
| F6 | Bukin | 0.6014451 | 0.995241 | 39.5723763 |
| F7 | Bohachevsky | 0.5983824 | 1.0317213 | 40.4998779 |
| F8 | Zakharov | 0.6149669 | 1.0155977 | 40.9370118 |
| F9 | Booth | 0.5762804 | 1.0061036 | 38.7605723 |
| F10 | Michalewicz | 0.7500612 | 0.9888142 | 47.4073201 |
| F11 | Eggholder | 0.6176157 | 0.9313977 | 39.4852418 |
| F12 | Himmelblau | 0.6145091 | 0.9157816 | 38.6021651 |

** results in seconds*

### *5.5. Comparative Evaluation*

To validate our work, two comparisons with state of the art methods were conducted. These state of the art methods used ten benchmark functions in the first comparison and seven benchmark functions in

the second comparison that were adopted in this research. Note that the results of comparative methods are selected from [111] and [112] for the first and second comparisons, respectively. In the first comparison, the comparative algorithms are Artificial Bee Colony (ABC) [9], Imperialist Competitive Algorithm (ICA) [113], Grey Wolf Optimizer (GWO) [18], Invasive Weed Optimization (IWO) [114], Genetic Algorithm (GA) [115], Particle Swarm Optimization (PSO) [116], differential Evolution (DE) [117], and Harmony Search (HS) [118, 119]. Where in the second comparison , the comparative algorithms are practical genetic algorithm (RGA) [120], gravitational search algorithm (GSA) [121], disruption GSA (D-GSA) [122], black hole GSA (BH-GSA) [123], clustered GSA (C-GSA) [124] and attractive repulsive GSA (AR-GSA) [125]. The parameter settings are shown in Table 9 as used by all comparative methods to allow a fair comparison.

Table 9: Experimental Configuration for First Comparison as in [111]

| **Parameter** | **Value** |
|---|---|
| Max Iterations ($L$) | 100 |
| Termination Criteria | $L$ |
| Population Size ($N$) | 20 |
| Decision Variables No. | based on benchmark function |
| No. of Runs | 31 |
| Lower Bound ($lb$) and Upper Bound ($ub$) | based on benchmark function |

The mean and standard deviation results of the first and second comparisons over 30 experimental runs are shown in Table 10 and Table 11 respectively.The best results are highlighted in the bold font. For the first comparison, it can be seen that the proposed method is able to achieve the best results for Sphere, Bukin, Bohachevsky, Zakharov, Booth, and Michalewicz functions. Also, for the uni-modal function with no local minima such as "Sphere" function outcome clarify that $\mathbf{SSA}_{GA-tuner}$ algorithm has the best result. On the other hand, the Emperor Penguins Colony (EPC) algorithm achieved the best results for Rastrigin, Ackley, and Griewank functions, with multi-local minima. Furthermore, $\mathbf{SSA}_{GA-tuner}$ obtains the best results for Michalewicz function which is complex and a multi-modal type. Based on the conducted experiments, the overall results confirm that the proposed $\mathbf{SSA}_{GA-tuner}$ algorithm is appropriate for optimization, whether the optimization problems subject has uni-modal or multi-modal search space nature. For the standard deviation results in the same table, it is notable that the proposed $\mathbf{SSA}_{GA-tuner}$ algorithm performance is stable.

For the second comparison, it can be seen that the proposed $\mathbf{SSA}_{GA-tuner}$ is able to achieve the best results for Sphere and Ackley functions only. Where the SSARM-SCA algorithm gets the best results in Rastrigin and Griewank functions. In addition, RGA and BH-GSA algorithms gets the best results in Rosenbrock function. Although the proposed algorithm did not get the best results in most of the functions, but its results were close to the competing algorithms. This comparison confirm that the proposed $\mathbf{SSA}_{GA-tuner}$ algorithm is appropriate for optimization, whether the optimization problems subject has uni-modal or multi-modal search space nature.

## 6. Conclusion

This paper proposes an enhanced version of the Salp Swarm Algorithm (SSA) for optimization problems. The enhancements of SSA involve the initial population diversity and the parameter control strategy. Firstly, the diversification of the Salp Swarm population is introduced to control the exploration aspects. Secondly, a new version of SSA referred as $\mathbf{SSA}_{GA-tuner}$ is proposed to enhance the parameters control of SSA using a self-adaptive parameter setting whereby genetic algorithm is adopted to find the optimal parameters for SSA at each generation.

Initially, the effect of the diversified population on the convergence behavior of $\mathbf{SSA}_{std}$ version is studied. The proposed algorithm is able to excel in the standard version of SSA in all benchmark functions. In conclusion, there is a positive impact of the diversified population on the performance of $\mathbf{SSA}_{std}$. In order to evaluate the impact of the self-adaptive parameter control on the convergence of $\mathbf{SSA}_{GA-tuner}$, the

Table 10: **Mean (avg)** and **Standard Deviation (std)** of best obtained results for employed benchmark functions

| Function | Statistical Measure | GA | ICA | PSO | ABC | DE |
|---|---|---|---|---|---|---|
| Sphere | avg | 2.80E-03 | 4.95E-06 | 9.04E-08 | 1.02E-02 | 1.34E-08 |
| | std | 4.80E-03 | 2.50E-05 | 1.30E-07 | 6.60E-03 | 1.45E-08 |
| Rastrigin | avg | 9.37E-01 | 1.21E+00 | 2.90E+00 | 1.08E+01 | 2.99E-01 |
| | std | 6.57E-01 | 1.40E+00 | 2.05E+00 | 3.08E+00 | 6.40E-01 |
| Ackley | avg | 9.07E-02 | 2.63E-04 | 6.09E-04 | 1.42E-01 | 1.77E-04 |
| | std | 7.68E-02 | 3.20E-04 | 4.68E-04 | 7.87E-02 | 8.09E-05 |
| Griewank | avg | 4.37E-02 | 2.64E-02 | 2.22E-02 | 1.33E-01 | 1.39E-02 |
| | std | 3.59E-02 | 1.25E-02 | 1.40E-02 | 4.10E-02 | 1.44E-02 |
| Rosenbrock | avg | 2.64E+00 | 3.39E+00 | 1.78E+00 | 1.55E+01 | 1.99E+00 |
| | std | 1.99E+00 | 6.88E+00 | 1.40E+00 | 7.82E+00 | 1.24E+00 |
| Bukin | avg | 1.17E+00 | 7.59E-02 | 2.30E-01 | 7.34E-01 | 8.87E-01 |
| | std | 2.76E+00 | 5.09E-02 | 1.05E-01 | 4.07E-01 | 5.17E-01 |
| Bohachevsky | avg | 8.50E-03 | 3.64E-13 | 4.97E-10 | 3.01E-07 | 8.45E-13 |
| | std | 3.58E-02 | 1.03E-12 | 9.82E-10 | 5.02E-07 | 1.49E-12 |
| Zakharov | avg | 2.23E+00 | 3.91E-01 | 4.70E-06 | 3.96E+00 | 1.83E-01 |
| | std | 3.69E+00 | 6.46E-01 | 9.76E-06 | 2.64E+00 | 1.50E-01 |
| Booth | avg | 5.82E-02 | 2.80E-03 | 8.49E-11 | 8.10E-06 | 2.80E-05 |
| | std | 1.22E-01 | 1.47E-02 | 1.38E-10 | 1.49E-05 | 4.36E-05 |
| Michalewicz | avg | -4.56E+00 | -4.57E+00 | -4.12E+00 | -2.83E+00 | -4.80E+00 |
| | std | 2.85E-01 | 3.13E-01 | 4.86E-01 | 2.11E-01 | 6.83E-02 |
| | | HS | IWO | GWO | EPC | $SSA_{GA-tuner}$ |
| Sphere | avg | 4.33E-01 | 9.21E-07 | 2.76E-12 | 3.32E-16 | **1.23E-44** |
| | std | 3.58E-01 | 5.21E-07 | 1.21E-11 | 1.36E-16 | **6.72E-44** |
| Rastrigin | avg | 7.38E+00 | 1.50E+01 | 2.64E+00 | **5.80E-14** | 2.01E+01 |
| | std | 2.23E+00 | 7.44E+00 | 3.74E+00 | **2.58E-14** | 8.05E+00 |
| Ackley | avg | 2.09E+00 | 1.90E-03 | 1.92E-05 | **3.18E-08** | 1.19E+00 |
| | std | 6.41E-01 | 5.74E-04 | 5.85E-05 | **6.78E-09** | 1.11E+00 |
| Griewank | avg | 4.44E-02 | 4.35E-02 | 5.42E-02 | **1.31E-02** | 1.94E-01 |
| | std | **1.14E-02** | 4.10E-02 | 1.70E-01 | 1.18E-02 | 1.44E-01 |
| Rosenbrock | avg | 6.95E+01 | 9.45E+00 | **1.75E+00** | 3.88E+00 | 2.69E+01 |
| | std | 6.33E+01 | 2.89E+01 | 2.04E+00 | **4.35E-02** | 5.02E+01 |
| Bukin | avg | 1.97E+00 | 2.86E-01 | 1.51E-01 | 9.56E-02 | **5.48E-02** |
| | std | 1.01E+00 | 1.25E-01 | 9.22E-02 | **2.50E-02** | 3.85E-02 |
| Bohachevsky | avg | 5.60E-03 | 2.00E-07 | 7.18E-10 | 1.11E-17 | **0.00E+00** |
| | std | 8.70E-03 | 2.86E-07 | 2.22E-09 | 4.47E-17 | **0.00E+00** |
| Zakharov | avg | 4.44E+00 | 2.49E-06 | 1.69E-08 | 5.49E-16 | **1.34E-23** |
| | std | 3.27E+00 | 1.39E-06 | 6.48E-08 | 1.82E-16 | **7.15E-23** |
| Booth | avg | 1.60E-02 | 2.76E-08 | 1.97E+00 | 7.34E-18 | **0.00E+00** |
| | std | 3.50E-02 | 2.66E-08 | 1.44E-01 | 6.47E-18 | **0.00E+00** |
| Michalewic | avg | -4.49E+00 | -3.94E+00 | -2.49E+00 | -1.80E+00 | **-7.05E+00** |
| | std | **1.60E-01** | 5.18E-01 | 4.40E-01 | 2.61E-01 | 9.32E-01 |

* *best results in bold*

Table 11: **Mean (avg)** and **Standard Deviation (std)** of best obtained results for employed benchmark functions

| Function | Statistical Measure | RGA | GSA | D-GSA | BH-GSA | C-GSA | AR-GSA | SSARM-SCA | $SSA_{GA-tuner}$ |
|---|---|---|---|---|---|---|---|---|---|
| Sphere | avg | 2.82E+02 | 7.58E-14 | 9.75E-01 | 3.33E-12 | 2.76E-13 | **0.0E+00** | **0.0E+00** | **1.23E-44** |
| | std | 3.16E+01 | 1.08E-13 | 1.97E-01 | 1.03E-12 | 9.44E-14 | **0.0E+00** | **0.0E+00** | **6.72E-44** |
| Rastrigin | avg | 1.44E+02 | 1.90E+02 | 1.87E+02 | 1.79E+01 | 1.87E+02 | 1.83E+01 | **1.74E+01** | 2.01E+01 |
| | std | 9.16E+00 | 2.35E+01 | 2.18E+01 | 5.21E+00 | 2.14E+01 | 4.47E+00 | **4.32E+00** | 8.05E+00 |
| Ackley | avg | 2.10E+01 | 2.10E+01 | 2.10E+01 | 2.10E+01 | 2.12E+01 | 2.09E+01 | 1.85E+01 | **1.19E+00** |
| | std | 4.67E-02 | 4.79E-02 | 5.29E-02 | 5.62E-02 | 1.59E-01 | 7.14E-02 | **4.54E-02** | 1.11E+00 |
| Griewank | avg | 5.91E+01 | 5.61E-03 | 1.57E+00 | 2.56E-03 | 7.39E-03 | 1.69E-03 | **1.53E-03** | 1.94E-01 |
| | std | 6.75E+00 | 6.39E-03 | 2.68E-01 | 5.05E-03 | 6.02E-03 | 3.83E-03 | **3.67E-03** | 1.44E-01 |
| Rosenbrock | avg | 1.13E+02 | 5.18E+01 | 7.36E+01 | **2.26E+01** | 5.13E+01 | 3.37E+01 | 3.15E+01 | 2.69E+01 |
| | std | **1.20E+01** | 2.51E+01 | 2.46E+01 | 2.68E+01 | 2.50E+01 | 2.73E+01 | 2.58E+01 | 5.02E+01 |

* *best results in bold*

comparative results against standard SSA and $\mathbf{SSA}_{std}$ show that the $\mathbf{SSA}_{GA-tuner}$ is able to yield the best results. Briefly, the results prove that the self-adaptive parameter control has a direct impact on the performance of the proposed SSA versions. In a nutshell, the proposed SSA versions are a very powerful enhancement that can be applied for a wide range of optimization problems.

Based on the experimental evaluation and verification carried out, it is notable that proposed methods tackle the exploration issue through increasing the population diversity, which in turn insure covering the entire search area as much as possible. In addition, the proposed methods tune the SSA algorithm to address the variation in different problems nature, so the algorithm become suitable to tackle any prediction problem.

Additionally, the experimental results confirm that the robustness of the diverse and parameter controlled algorithm that develops an optimal set of weights and biases values for the BPNN predictor added an edge to the prediction process, in addition to build a parameter-less optimization algorithm and to make use of the full utilization of the algorithm efficiency by striking the right balance among wide-area exploration and local-nearby exploitation during the search, in order to helped enhance the BPNN's performance.

As the proposed SSA versions reveal very successful outcomes, in the future, the proposed SSA versions can be adapted for combinatorial optimization problems such as scheduling problems. Furthermore, other ways of parameter tuning such as control parameter tuning and adaptive parameter control strategies can be investigated. Other enhancement in the SSA can be studied such as adapting structural population methods and fusing natural selection principles.

## Acknowledgment

This work is supported by University Kebangsaan Malaysia (DIP-2016-024)

## Compliance with Ethical Standards

This manuscript is the authors' original work and has not been published nor has it been submitted simultaneously elsewhere. All authors have checked the manuscript and have agreed to the submission.

## References

[1] H. Faris, A.-Z. Ala'M, A. A. Heidari, I. Aljarah, M. Mafarja, M. A. Hassonah, H. Fujita, An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks, Information Fusion 48 (2019) 67–83.

[2] M. Mafarja, I. Aljarah, A. A. Heidari, A. I. Hammouri, H. Faris, A.-Z. Ala'M, S. Mirjalili, Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems, Knowledge-Based Systems 145 (2018) 25–45.

[3] M. Mafarja, I. Aljarah, A. A. Heidari, H. Faris, P. Fournier-Viger, X. Li, S. Mirjalili, Binary dragonfly optimization for feature selection using time-varying transfer functions, Knowledge-Based Systems 161 (2018) 185–204.

[4] A. A. Heidari, P. Pahlavani, An efficient modified grey wolf optimizer with lévy flight for optimization tasks, Applied Soft Computing 60 (2017) 115–134.
[5] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in engineering software 95 (2016) 51–67.
[6] H. Chen, Y. Xu, M. Wang, X. Zhao, A balanced whale optimization algorithm for constrained engineering design problems, Applied Mathematical Modelling 71 (2019) 45–59.
[7] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: theory and application, Advances in Engineering Software 105 (2017) 30–47.
[8] J. Luo, H. Chen, Y. Xu, H. Huang, X. Zhao, et al., An improved grasshopper optimization algorithm with application to financial stress prediction, Applied Mathematical Modelling 64 (2018) 654–668.
[9] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, Journal of global optimization 39 (3) (2007) 459–471.
[10] S. K. Nseef, S. Abdullah, A. Turky, G. Kendall, An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems, Knowledge-based systems 104 (2016) 14–23.
[11] S. Abdullah, S. K. Nseef, A. Turky, An interleaved artificial bee colony algorithm for dynamic optimisation problems, Connection Science 30 (3) (2018) 272–284.
[12] A. Colorni, M. Dorigo, V. Maniezzo, et al., Distributed optimization by ant colonies, in: Proceedings of the first European conference on artificial life, Vol. 142, Cambridge, MA, 1992, pp. 134–142.
[13] R. Eberhart, J. Kennedy, Particle swarm optimization, in: Proceedings of the IEEE international conference on neural networks, Vol. 4, Citeseer, 1995, pp. 1942–1948.
[14] M. Alweshah, A. I. Hammouri, S. Tedmori, Biogeography-based optimisation for data classification problems, International Journal of Data Mining, Modelling and Management 9 (2) (2017) 142–162.
[15] M. Alweshah, Construction biogeography-based optimization algorithm for solving classification problems, Neural Computing and Applications (2018) 1–10.
[16] K. M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE control systems magazine 22 (3) (2002) 52–67.
[17] Z. Cai, J. Gu, C. Wen, D. Zhao, C. Huang, H. Huang, C. Tong, J. Li, H. Chen, An intelligent parkinson's disease diagnostic system based on a chaotic bacterial foraging optimization enhanced fuzzy knn approach, Computational and mathematical methods in medicine (2018) 24.
[18] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in engineering software 69 (2014) 46–61.
[19] H. Al Nsour, M. Alweshah, A. I. Hammouri, H. Al Ofeishat, S. Mirjalili, A hybrid grey wolf optimiser algorithm for solving time series classification problems, Journal of Intelligent Systems 29 (1) (2018) 846–857.
[20] X. Zhao, X. Zhang, Z. Cai, X. Tian, X. Wang, Y. Huang, H. Chen, L. Hu, Chaos enhanced grey wolf optimization wrapped elm for diagnosis of paraquat-poisoned patients, Computational biology and chemistry 78 (2019) 481–490.
[21] W.-T. Pan, A new fruit fly optimization algorithm: taking the financial distress model as an example, Knowledge-Based Systems 26 (2012) 69–74.
[22] L. Shen, H. Chen, Z. Yu, W. Kang, B. Zhang, H. Li, B. Yang, D. Liu, Evolving support vector machines using fruit fly optimization for medical data classification, Knowledge-Based Systems 96 (2016) 61–75.
[23] N. S. Jaddi, J. Alvankarian, S. Abdullah, Kidney-inspired algorithm for optimization problems, Communications in Nonlinear Science and Numerical Simulation 42 (2017) 358–369.
[24] N. S. Jaddi, S. Abdullah, Kidney-inspired algorithm with reduced functionality treatment for classification and time series prediction, PloS one 14 (1) (2019) e0208308.
[25] M. Alweshah, Firefly algorithm with artificial neural network for time series problems, Research Journal of Applied Sciences, Engineering and Technology 7 (19) (2014) 3978–3982.
[26] M. Alweshah, S. Abdullah, Hybridizing firefly algorithms with a probabilistic neural network for solving classification problems, Applied Soft Computing 35 (2015) 513–524.
[27] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, Future Generation Computer Systems 97 (2019) 849–872.
[28] A. Rahiminasab, P. Tirandazi, M. Ebadi, A. Ahmadian, M. Salimi, An energy-aware method for selecting cluster heads in wireless sensor networks, Applied Sciences 10 (21) (2020) 7886.
[29] Z.-Y. He, A. Abbes, H. Jahanshahi, N. D. Alotaibi, Y. Wang, Fractional-order discrete-time sir epidemic model with vaccination: Chaos and complexity, Mathematics 10 (2) (2022) 165.
[30] M. Tubishat, N. Idris, L. Shuib, M. A. Abushariah, S. Mirjalili, Improved salp swarm algorithm based on opposition based learning and novel local search algorithm for feature selection, Expert Systems with Applications 145 (2020) 113122.
[31] T. Sun, M.-h. Xu, A swarm optimization genetic algorithm based on quantum-behaved particle swarm optimization, Computational intelligence and neuroscience (2017) 1–15.
[32] W. Zhang, W. Hou, D. Yang, Z. Xing, M. Gen, Multiobjective pso algorithm with multi-directional convergence strategy to solve flow shop scheduling problems, in: International Conference on Management Science and Engineering Management, Springer, 2019, pp. 750–759.
[33] S. L. Edathil, S. P. Singh, Aco and cs-based hybrid optimisation method for optimum sizing of the shes, IET Renewable Power Generation 13 (10) (2019) 1789–1801.
[34] F. Heydarpour, E. Abbasi, M. Ebadi, S.-M. Karbassi, Solving an optimal control problem of cancer treatment by artificial neural networks., International Journal of Interactive Multimedia & Artificial Intelligence 6 (4) (2020).
[35] T.-H. Zhao, M. I. Khan, Y.-M. Chu, Artificial neural networking (ann) analysis for heat and entropy generation in flow of non-newtonian fluid between two rotating disks, Mathematical Methods in the Applied Sciences (2021).
[36] F. Heydarpoor, S. M. Karbassi, N. Bidabadi, M. J. Ebadi, Solving multi-objective functions for cancer treatment by

using metaheuristic algorithms, algorithms 21 (2020) 22.
[37] D. H. Wolpert, W. G. Macready, et al., No free lunch theorems for optimization, IEEE transactions on evolutionary computation 1 (1) (1997) 67–82.
[38] 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, Advances in Engineering Software 114 (2017) 163–191.
[39] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A.-Z. Ala'M, S. Mirjalili, H. Fujita, An efficient binary salp swarm algorithm with crossover scheme for feature selection problems, Knowledge-Based Systems 154 (2018) 43–67.
[40] R. A. Ibrahim, A. A. Ewees, D. Oliva, M. A. Elaziz, S. Lu, Improved salp swarm algorithm based on particle swarm optimization for feature selection, Journal of Ambient Intelligence and Humanized Computing (2018) 1–15.
[41] M. Khamees, A. Albakry, K. Shaker, Multi-objective feature selection: Hybrid of salp swarm and simulated annealing approach, in: International Conference on New Trends in Information and Communications Technology Applications, Springer, 2018, pp. 129–142.
[42] G. I. Sayed, G. Khoriba, M. H. Haggag, A novel chaotic salp swarm algorithm for global optimization and feature selection, Applied Intelligence 48 (10) (2018) 3462–3481.
[43] A. A. El-Fergany, Extracting optimal parameters of pem fuel cells using salp swarm optimizer, Renewable Energy 119 (2018) 641–648.
[44] S. Asaithambi, M. Rajappa, Swarm intelligence-based approach for optimal design of cmos differential amplifier and comparator circuit using a hybrid salp swarm algorithm, Review of Scientific Instruments 89 (5) (2018) 054702.
[45] S. Ekinci, B. Hekimoglu, Parameter optimization of power system stabilizer via salp swarm algorithm, in: 2018 5th International Conference on Electrical and Electronic Engineering (ICEEE), IEEE, 2018, pp. 143–147.
[46] A. G. Hussien, A. E. Hassanien, E. H. Houssein, Swarming behaviour of salps algorithm for predicting chemical compound activities, in: 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), IEEE, 2017, pp. 315–320.
[47] J. Wang, Y. Gao, X. Chen, A novel hybrid interval prediction approach based on modified lower upper bound estimation in combination with multi-objective salp swarm algorithm for short-term load forecasting, Energies 11 (6) (2018) 1561.
[48] A. Ibrahim, A. Ahmed, S. Hussein, A. E. Hassanien, Fish image segmentation using salp swarm algorithm, in: International Conference on Advanced Machine Learning Technologies and Applications, Springer, 2018, pp. 42–51.
[49] J. Zhang, Z. Wang, X. Luo, Parameter estimation for soil water retention curve using the salp swarm algorithm, Water 10 (6) (2018) 815.
[50] R. Abbassi, A. Abbassi, A. A. Heidari, S. Mirjalili, An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models, Energy Conversion and Management 179 (2019) 362–372.
[51] A. K. Barik, D. C. Das, Active power management of isolated renewable microgrid generating power from rooftop solar arrays, sewage waters and solid urban wastes of a smart city using salp swarm algorithm, in: 2018 Technologies for Smart-City Energy Security and Power (ICSESP), IEEE, 2018, pp. 1–6.
[52] D. Guha, P. Roy, S. Banerjee, A maiden application of salp swarm algorithm optimized cascade tilt-integral-derivative controller for load frequency control of power systems, IET Generation, Transmission and Distribution (oct 2018). doi:10.1049/iet-gtd.2018.6100.
[53] N. Singh, F. Chiclana, J.-P. Magnot, et al., A new fusion of salp swarm with sine cosine for optimization of non-linear functions, Engineering with Computers (2019) 1–28.
[54] Q. Zhang, H. Chen, A. A. Heidari, X. Zhao, Y. Xu, P. Wang, Y. Li, C. Li, Chaos-induced and mutation-driven schemes boosting salp chains-inspired optimizers, Ieee Access 7 (2019) 31243–31261.
[55] A. E. Hegazy, M. Makhlouf, G. S. El-Tawel, Improved salp swarm algorithm for feature selection, Journal of King Saud University-Computer and Information Sciences 32 (3) (2020) 335–344.
[56] R. M. Rizk-Allah, A. E. Hassanien, M. Elhoseny, M. Gunasekaran, A new binary salp swarm algorithm: development and application for optimization tasks, Neural Computing and Applications (2018) 1–23.
[57] H. Zhang, Z. Cai, X. Ye, M. Wang, F. Kuang, H. Chen, C. Li, Y. Li, A multi-strategy enhanced salp swarm algorithm for global optimization, Engineering with Computers (2020) 1–27.
[58] J. Wu, R. Nan, L. Chen, Improved salp swarm algorithm based on weight factor and adaptive mutation, Journal of Experimental & Theoretical Artificial Intelligence 31 (3) (2019) 493–515.
[59] L. Zhang, C. Li, Y. Wu, J. Huang, Z. Cui, An improved salp swarm algorithm with spiral flight search for optimizing hybrid active power filters' parameters, IEEE Access (2020).
[60] Y. Yin, Q. Tu, X. Chen, Enhanced salp swarm algorithm based on random walk and its application to training feedforward neural networks, Soft Computing (2020) 1–17.
[61] M. Mao, H. Huang, L. Zhang, B. Chong, L. Zhou, Maximum power exploitation for grid-connected pv system under fast-varying solar irradiation levels with modified salp swarm algorithm, Journal of Cleaner Production (2020) 122158.
[62] S. S. Alresheedi, S. Lu, M. Abd Elaziz, A. A. Ewees, Improved multiobjective salp swarm optimization for virtual machine placement in cloud computing, Human-centric Computing and Information Sciences 9 (1) (2019) 15.
[63] J. Zhang, J. Wang, Improved salp swarm algorithm based on levy flight and sine cosine operator, IEEE Access (2020).
[64] P. Chen, C. You, P. Ding, Event classification using improved salp swarm algorithm based probabilistic neural network in fiber-optic perimeter intrusion detection system, Optical Fiber Technology 56 (2020) 102182.
[65] B. Ma, H. Ni, X. Zhu, R. Zhao, A comprehensive improved salp swarm algorithm on redundant container deployment problem, IEEE Access 7 (2019) 136452–136470.
[66] A. A. Ateya, A. Muthanna, A. Vybornova, A. D. Algarni, A. Abuarqoub, Y. Koucheryavy, A. Koucheryavy, Chaotic salp swarm algorithm for sdn multi-controller networks, Engineering Science and Technology, an International Journal 22 (4) (2019) 1001–1012.

[67] F. Mohanty, S. Rup, B. Dash, B. Majhi, M. Swamy, An improved scheme for digital mammogram classification using weighted chaotic salp swarm algorithm-based kernel extreme learning machine, Applied Soft Computing (2020) 106266.
[68] X. Zhao, F. Yang, Y. Han, Y. Cui, An opposition-based chaotic salp swarm algorithm for global optimization, IEEE Access 8 (2020) 36485–36501.
[69] A. Ibrahim, S. Mohammed, H. A. Ali, S. E. Hussein, Breast cancer segmentation from thermal images based on chaotic salp swarm algorithm, IEEE Access 8 (2020) 122121–122134.
[70] A. Altan, S. Karasu, Recognition of covid-19 disease from x-ray images by hybrid model consisting of 2d curvelet transform, chaotic salp swarm algorithm and deep learning technique, Chaos, Solitons & Fractals (2020) 110071.
[71] W. H. El-Ashmawi, A. F. Ali, A modified salp swarm algorithm for task assignment problem, Applied Soft Computing (2020) 106445.
[72] Y. Yu, H. Wang, N. Li, H. Zhang, Z. Su, X. Shao, Finite-time model-assisted active disturbance rejection control with a novel parameters optimizer for hypersonic reentry vehicle subject to multiple disturbances, Aerospace Science and Technology 79 (2018) 588–600.
[73] R. Senkerik, A. Viktorin, M. Pluhacek, T. Kadavy, I. Zelinka, How unconventional chaotic pseudo-random generators influence population diversity in differential evolution, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2018, pp. 524–535.
[74] S. Dash, S. Dey, A. Augustine, R. S. Dhar, J. Pidanic, Z. Nemec, G. Trivedi, Riveropt: A multiobjective optimization framework based on modified river formation dynamics heuristic, in: 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), IEEE, 2019, pp. 233–238.
[75] E.-G. Talbi, Metaheuristics: from design to implementation, Vol. 74, John Wiley & Sons, 2009.
[76] T. M. Blackwell, P. Bentley, Don't push me! collision-avoiding swarms, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), Vol. 2, IEEE, 2002, pp. 1691–1696.
[77] Z. Song, B. Liu, H. Cheng, Adaptive particle swarm optimization with population diversity control and its application in tandem blade optimization, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 233 (6) (2019) 1859–1875.
[78] X.-F. Xie, W.-J. Zhang, Z.-L. Yang, Dissipative particle swarm optimization, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), Vol. 2, IEEE, 2002, pp. 1456–1461.
[79] M. I. Menhas, M. Fei, L. Wang, X. Fu, A novel hybrid binary pso algorithm, in: International Conference in Swarm Intelligence, Springer, 2011, pp. 93–100.
[80] R. Brits, A. P. Engelbrecht, F. Van den Bergh, A niching particle swarm optimizer, in: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, Vol. 2, Singapore: Orchid Country Club, 2002, pp. 692–696.
[81] L. Huang, C.-T. Ng, A. H. Sheikh, M. C. Griffith, Niching particle swarm optimization techniques for multimodal buckling maximization of composite laminates, Applied Soft Computing 57 (2017) 495–503.
[82] Y. Li, Y. Chen, J. Zhong, Z. Huang, Niching particle swarm optimization with equilibrium factor for multi-modal optimization, Information Sciences 494 (2019) 233–246.
[83] X. Li, Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, in: Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 105–116.
[84] P. A. Wilhelm, Pheromone particle swarm optimization of stochastic systems, Ph.D. thesis (2008).
[85] M. Kovaleva, B. A. Zeb, D. Bulger, K. P. Esselle, Radiation performance enhancement of a compact fabry-perot cavity antenna using particle swarm optimization, in: 2015 International Symposium on Antennas and Propagation (ISAP), IEEE, 2015, pp. 1–3.
[86] A. Lin, W. Sun, H. Yu, G. Wu, H. Tang, Global genetic learning particle swarm optimization with diversity enhancement by ring topology, Swarm and evolutionary computation 44 (2019) 571–583.
[87] F. Blanquart, Evolutionary epidemiology models to predict the dynamics of antibiotic resistance, Evolutionary applications 12 (3) (2019) 365–383.
[88] Zhang, Y. Yu, S. Zheng, Y. Todo, S. Gao, Exploitation enhanced sine cosine algorithm with compromised population diversity for optimization, in: 2018 IEEE International Conference on Progress in Informatics and Computing (PIC), IEEE, 2018, pp. 1–7.
[89] W. Deng, J. Xu, H. Zhao, An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem, IEEE Access 7 (2019) 20281–20292.
[90] L. Eskandari, A. Jafarian, P. Rahimloo, D. Baleanu, A modified and enhanced ant colony optimization algorithm for traveling salesman problem, in: Mathematical Methods in Engineering, Springer, 2019, pp. 257–265.
[91] U. Balande, D. Shrimankar, Srifa: Stochastic ranking with improved-firefly-algorithm for constrained optimization engineering design problems, Mathematics 7 (3) (2019) 250.
[92] Á. E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on evolutionary computation 3 (2) (1999) 124–141.
[93] C. Huang, Y. Li, X. Yao, A survey of automatic parameter tuning methods for metaheuristics, IEEE Transactions on Evolutionary Computation (2019).
[94] A. Eiben, J. Smith, Parameter control, in: Introduction to Evolutionary Computing, Springer, 2015, pp. 131–146.
[95] E. Mezura-Montes, Deterministic parameter control in differential evolution with combined variants for constrained search spaces, Numerical and Evolutionary Optimization–NEO 2017 785 (2019) 3.
[96] T. Kuno, Deterministic parameter selection of artificial bee colony based on diagonalization, in: Hybrid Intelligent Systems: 18th International Conference on Hybrid Intelligent Systems (HIS 2018) Held in Porto, Portugal, December 13-15, 2018, Vol. 923, Springer, 2019, p. 85.
[97] G. Bernstein, K. O'Brien, Stochastic agent-based simulations of social networks, in: Proceedings of the 46th annual

simulation symposium, Society for Computer Simulation International, 2013, p. 5.
[98] G. Sun, Y. Lan, R. Zhao, Differential evolution with gaussian mutation and dynamic parameter adjustment, Soft Computing 23 (5) (2019) 1615–1642.
[99] W.-J. Yu, M. Shen, W.-N. Chen, Z.-H. Zhan, Y.-J. Gong, Y. Lin, O. Liu, J. Zhang, Differential evolution with two-level parameter adaptation, IEEE Transactions on Cybernetics 44 (7) (2014) 1080–1099.
[100] W. Zhu, Y. Tang, J.-A. Fang, W. Zhang, Adaptive population tuning scheme for differential evolution, Information Sciences 223 (2013) 164–191.
[101] M. Kavoosi, M. A. Dulebenets, O. F. Abioye, J. Pasha, H. Wang, H. Chi, An augmented self-adaptive parameter control in evolutionary computation: A case study for the berth scheduling problem, Advanced Engineering Informatics 42 (2019) 100972.
[102] X. Chen, F. Kopsaftopoulos, Q. Wu, H. Ren, F.-K. Chang, A self-adaptive 1d convolutional neural network for flight-state identification, Sensors 19 (2) (2019) 275.
[103] A. El Afia, O. Aoun, S. Garcia, Adaptive cooperation of multi-swarm particle swarm optimizer-based hidden markov model, Progress in Artificial Intelligence (2019) 1–12.
[104] A. N. Haiman, Creature feature: Salps (2015).
URL `https://theethogram.com/2015/05/04/featured-creature-salps`
[105] H. Holland John, Adaptation in natural and artificial systems, Ann Arbor: University of Michigan Press (1975).
[106] D. E. Goldberg, J. H. Holland, Genetic algorithms and machine learning, Machine learning 3 (2) (1988) 95–99.
[107] J. H. Holland, Genetic algorithms, Scientific american 267 (1) (1992) 66–73.
[108] K. Premalatha, A. Natarajan, Hybrid pso and ga for global maximization, Int. J. Open Problems Compt. Math 2 (4) (2009) 597–608.
[109] N. Ghorbani, A. Kasaeian, A. Toopshekan, L. Bahrami, A. Maghami, Optimizing a hybrid wind-pv-battery system using ga-pso and mopso for reducing cost and increasing reliability, Energy 154 (2018) 581–591.
[110] S. Mirjalili, Genetic algorithm, in: Evolutionary Algorithms and Neural Networks, Springer, 2019, pp. 43–55.
[111] S. Harifi, M. Khalilian, J. Mohammadzadeh, S. Ebrahimnejad, Emperor penguins colony: a new metaheuristic algorithm for optimization, Evolutionary Intelligence (2019) 1–16.
[112] M. Zivkovic, C. Stoean, A. Chhabra, N. Budimirovic, A. Petrovic, N. Bacanin, Novel improved salp swarm algorithm: An application for feature selection, Sensors 22 (5) (2022) 1711.
[113] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE congress on evolutionary computation, IEEE, 2007, pp. 4661–4667.
[114] A. R. Mehrabian, C. Lucas, A novel numerical optimization algorithm inspired from weed colonization, Ecological informatics 1 (4) (2006) 355–366.
[115] S. Sivanandam, S. Deepa, Introduction to genetic algorithms, Springer Science & Business Media, 2007.
[116] J. Kennedy, Particle swarm optimization, Springer, US, 2017, p. 760–766.
[117] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.
[118] Z. W. Geem, J. H. Kim, G. V. Loganathan, A new heuristic optimization algorithm: harmony search, simulation 76 (2) (2001) 60–68.
[119] A. Turky, S. Abdullah, A. Dawod, A dual-population multi operators harmony search algorithm for dynamic optimization problems, Computers & Industrial Engineering 117 (2018) 19–28.
[120] C. W. Ahn, Practical genetic algorithms, Advances in Evolutionary Algorithms: Theory, Design and Practice (2006) 7–22.
[121] E. Rashedi, H. Nezamabadi-pour, Improving the precision of cbir systems by feature selection using binary gravitational search algorithm, in: The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012), IEEE, 2012, pp. 039–042.
[122] R. H. Ginardi, A. Izzah, A new operator in gravitational search algorithm based on the law of momentum, in: International Conference on Information, Communication Technology and System, 2014, pp. 105–110.
[123] M. Doraghinejad, H. Nezamabadi-Pour, Black hole: a new operator for gravitational search algorithm, International Journal of Computational Intelligence Systems 7 (5) (2014) 809–826.
[124] M. Shams, E. Rashedi, A. Hakimi, Clustered-gravitational search algorithm and its application in parameter optimization of a low noise amplifier, Applied Mathematics and Computation 258 (2015) 436–453.
[125] H. Zandevakili, E. Rashedi, A. Mahani, Gravitational search algorithm with both attractive and repulsive forces, Soft Computing 23 (3) (2019) 783–825.