Self-adaptive salp swarm algorithm for optimization problems

Sofian Kassaymeh1,2 · Salwani Abdullah2 · Mohammed Azmi Al-Betar3,4 · Mohammed Alweshah5,6 · Mohamad Al-Laham7 · Zalinda Othman2

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 SSAstd, (ii) SSA parameters are tuned using a self-adaptive technique-based genetic algorithm (GA) referred as SSA_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 SSAstd enhances convergence behavior, and self-adaptive parameter tuning of SSA_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 (Faris et al. 2019; Mafarja et al. 2018, ?; Heidari and Pahlevani 2017). The recent swarm-based intelligence algorithms are Whale Optimization Algorithm (WOA) (Mirjalili and Lewis 2016; Chen et al. 2019), Grasshopper Optimization Algorithm (GOA) (Saremi et al. 2017; Luo et al. 2018), Artificial Bee Colony (ABC) (Karaboga and Basturk 2007; Nseef et al. 2016; Abdullah et al. 2018), Ant Colony Optimization (ACO) (Coloni et al. 1992), Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995), Biogeography-Based Algorithm (Alweshah et al. 2017; Alweshah 2018), Bacterial Foraging Optimization (BFO) (Passino 2002; Cai et al. 2018), Grey Wolf Optimizer (GWO) (Mirjalili et al. 2014; Al Nsour et al. 2018; Zhao et al. 2019), Fruit Fly Optimization (FOA) (Pan 2012; Shen et al. 2016), Kidney-Inspired Algorithm (KA)(Jaddi et al. 2017; Jaddi and Abdullah 2019), Firefly Algorithm (Alweshah 2014; Alweshah and Abdullah 2015), and Harris Hawk Optimizer (HHO) (Heidari et al. 2019). 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 (Rahiminasab et al. 2020; He et al. 2022).
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. Therefore, the theoretical aspects of the solvent algorithm could 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 (Sun and Xu 2017), at the same time the PSO converges easy but ACO converges slowly (Zhang et al. 2019). Moreover, exploration and exploitation for ABC and BFO are well-organized and have better capability, whereas it is poor in for ACO (Edathil and Singh 2019). 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 (Sun and Xu 2017, Heydarpoor et al. 2020).

Based on “No Free Lunch” (NFL) theorem (Wolpert and Macready 1997), 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 Mirjalili et al. (2017) 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 (Mirjalili et al. 2017). So, it has been applied to a lot of variety of optimization problems. For instance in (Faris et al. 2018; Ibrahim et al. 2018; Khamees et al. 2018; Sayed et al. 2018), the SSA algorithm is employed for Feature Selection Problem, whereas in El-Fergany (2018), 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 Asaithambi and Rajappa (2018). Also, Ekinci and Hekimoglu (2018) employs the SSA algorithm in calibrating the power system stabilizer for a power system for multi-machine. Next, the SSA is then used by Hussien et al. (2017) to estimate the activities of a chemical substance. Furthermore, in Wang et al. (2018), the study of Short-Term Load Forecasting utilizing the SSA classifier was conducted. Also, the problem of fish image segmentation utilizing SSA in Ibrahim et al. (2018). Moreover, SSA employed for predicting parameter values for the curve of soil water retention which proposed in Zhang et al. (2018), where SSA is proposed for parameter optimization of a detection model used for photovoltaic cell techniques (Abbassi et al. 2019). Furthermore, utilizing SSA for power load frequency control and for load frequency control of power systems was done in Barik and Das (2018) and Holland John (2018) 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, (Singh et al. 2019; Sayed et al. 2018; Zhang et al. 2019; Hegazy et al. 2020; Rizk-Allah et al. 2018; Sayed et al. 2018; Zhang et al. 2019), bad convergence accuracy (Zhang et al. 2020; Wu et al. 2019, Ibrahim et al. (2018)), slow convergence rate (Zhang et al. 2020; Wu et al. 2019; Yin et al. 2020; Mao et al. 2020; Alresheedi et al. 2019; Zhang and Wang 2018; Chen et al. 2020; Ma et al. 2019; Zhang et al. 2019; Sayed et al. 2018; Ateya et al. 2019; Mohanty et al. 2020; Zhao et al. 2020; Ibrahim et al. 2020; Altan and Karasu 2020). Lack of randomization components, (Yin et al. 2020), problems in discrete domain (El-Ashmawi and Ali 2020), deficiency of optimization ability, (Ma et al. 2019), and exploration ability (Zhang et al. 2019).

A few improvements attempts on SSA were proposed in the last years. Hegazy et al. (2020) enhances the SSA structure through tuning control parameters. In addition, a binary SSA algorithm was introduced to tackle the Arctan transformation problem (Rizk-Allah et al. 2018). Also, a chaos-induced SSA is proposed in Yu et al. (2018) 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 (Zhang et al. 2019; Sayed et al. 2018).

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 (SSAstd).
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 (SSA\textsubscript{GA-tuner}).

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.

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

The rest of the paper is organized as follows: Sect. 2 scans the Related Works. The fundamental background for the standard SSA and GA are discussed in Sect. 3. The proposed diversified SSA algorithm and self-adaptive SSA algorithms are described in Sect. 4. Results and discussions are thoroughly discussed and analyzed in Sect. 5. Finally, the conclusion and possible future directions are illustrated in Sect. 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 Sect. 2.1 while the proper and relevant literature about control parameters are given in Sect. 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) (Senkerik et al. 2018; Dash et al. 2019). 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 (Talbi 2009; Dash et al. 2019).

#### 2.2 Parameter control strategies

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.

All the mentioned methods focus on increasing the initial population diversity in order to increases the robustness of the proposed algorithm toward premature convergence (Song et al. 2019; Zhang et al. 2018; Deng et al. 2019), not trapped in local optima (Eskandari et al. 2019), and achieve balance among exploitation and exploration (Balande and Shrimankar 2019).

| Study                          | What proposed to achieve population diversity |
|-------------------------------|-----------------------------------------------|
| Blackwell and Bentley (2002); Song et al. (2019) | Avoiding excessive gathering in promising optimal areas |
| Xie et al. (2002), Menhas et al. (2011) | Avoiding premature convergence by use the dissipation method |
| Brits et al. (2002), Huang et al. (2017), Li et al. (2019) | Utilized niching techniques to accelerate the convergence |
| Li (2004), Wilhelm (2008), Kovaleva et al. (2015) | Adaptively calibrate swarm number |
| Lin et al. (2019), Blanquart (2019) | Mutation method based individual level |

Normally, the parameter settings of any optimization algorithm can be classified into two types: parameter tuning and parameter control (Eiben et al. 1999). 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.

There are three types of parameter control strategies: deterministic, adaptive and self-adaptive illustrated in Fig. 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.

3 Research background

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

3.1 Fundamentals to salp swarm algorithm

Salp swarm algorithm (SSA) was proposed by Mirjalili et al. (2017). 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 Fig. 2a. In order to move in the sea, salps are able to form a chain, namely “swarm chain” shown also in Fig. 2b. 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 \( d\text{im} - \text{dimension} \) in the search area, where \( \text{dim} \) 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)

\[
x^i = \text{rand}(x_j^i) \ast (ub - lb) + lb
\]  

(1)

where every solution is represented as \( x^i \) vector where \( i \in (1, 2, \ldots, n) \) in which \( n \) is the population size, \( j \in (1, 2, \ldots, \text{dim}) \). Each decision variable \( x^i_j \in [lb_j, ub_j] \) 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 Fig. 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).

\[
x_j^L = \begin{cases} F_j + c_1((ub_j - lb_j)c_2 + lb_j) & c_3 \geq 0.5 \\ F_j - c_1((ub_j - lb_j)c_2 + lb_j) & c_3 < 0.5 \end{cases}
\]  

(2)

where \( x_j^L \) represent the salp leader coordinates in the \( j^{\text{th}} \) dimension, \( F_j \) is the food source coordinates in the \( j^{\text{th}} \) dimen-
Self-adaptive salp swarm algorithm...

Fig. 2  a Single salp, b Salps chain (salps swarm) Haiman (2015)

Fig. 3  Standard SSA algorithm population representation

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

\[ x^i_j = \frac{1}{2} (x^i_j + x^{i-1}_j) \]  \hspace{1cm} (4)

where \( i \geq 2 \) and \( x^i_j \) 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 Fig. 4, and the pseudo-code is given in Algorithm 1.

\[ c_1 = 2e^{-(\frac{l}{L})^2} \]  \hspace{1cm} (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].
3.2 Fundamentals to genetic algorithm

Genetic algorithm (GA) is a popular population-based evolutionary-based algorithm proposed by Wang et al. (2018). 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 (Goldberg and Holland 1988; Holland 1992). 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 Fig. 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 (Premalatha and Natarajan 2009; Ghorbani et al. 2018; Mirjalili 2019), 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,

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

1: --- Stage 1: Initialization: Random Population ---
2: initialize a random initial population as Popinit using Eq. (5)
3: calculate the initial fitness of all solutions in Popinit
4: find the best solution referred as Solbest
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 \( c_l \) using Eq. (3)
10: for each solution in Popinit do
11: update the first salp using Eq. (2)
12: update the remaining salps using Eq. (4)
13: end for
14: update Popinit referred as Poptemp
15: for each salp in each solution in Poptemp do
16: if \( x_j > ub \) then
17: \( x_j = ub \)
18: else if \( x_j < lb \) then
19: \( x_j = lb \)
20: end if
21: end for
22: update Poptemp referred as Popnew
23: calculate the fitness of all solutions in the updated population (Popnew)
24: \( l++ \)
25: end while
26: select best solution in Popnew referred as Solnew
27: if Solnew is better than Solbest then
28: Solbest = Solnew
29: end if
30: return Solbest
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, 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 Fig. 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 Fig. 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 Fig. 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.

---

**Algorithm 2 Standard GA Algorithm Pseudo Code**

1: START
2: Initialize random population
3: Calculate fitness for all solutions in population
4: Compute fitness
5: for each solution in population do
6: Selection
7: Crossover
8: Mutation
9: Calculate fitness
10: UNTIL population has converged
11: end for
12: STOP
13: Return best solution

---

### 4 Proposed method

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

(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 $\text{SSA}_{std}$.

(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 $\text{SSA}_{GA-tuner}$.

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 $\text{SSA}_{std}$.

Initially, multiple random populations as many as $(\text{Max #Pop})$ are generated using Eq. (5). This done in a loop of $k$ cycles as shown in Fig. 8. At each $k$ cycle (say $i$), the standard deviation of the generated population $\text{std}(X_k)$ is calculated. To elaborate, for every decision variable $(x^i_j)(k)$ in the population $k$ the standard deviation is calculated using Eq. (6), and the average of standard deviations (i.e., $\text{avg}(\text{std}(x^i_j)_{(i,j)})$ 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.

$$x^i = \text{rand}(x^i_j) \times (ub - lb) + lb$$

$x^i$ represents 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.

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

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

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

$x^i$ represents the $i^{th}$ generated solution. In addition, $\overline{x}$ represents the average of the generated solution. Also, the $n$ represents the population size.
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 SSA_GA-tuner. In SSA_GA-tuner, GA plays a crucial role in determining the optimal parameters for SSA.

The proposed SSA_GA-tuner has five steps as shown in the flowchart visualized in Fig. 10 and pseudo-coded in Algorithm 5. These steps can be thoroughly discussed as follows:

**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 \( (k \leq \text{Max#Pop}) \) do
5. calculate standard deviation for every decision variable \( (x_j^i)_{|j|} \) in \( \text{Pop}_{\text{ini}} \) referred as \( \text{std}(x_j^i)_{|j|} \) using Eq.(6)
6. calculate the average of standard deviations \( \text{avg}(\text{std}(x_j^i)_{|j|}) \) of the decision variables using Eq.(7)
7. \( k++ \)
8. end while
9. select the best population based on highest value of \( \text{avg}(\text{std}(x_j^i)_{|j|}) \) referred as \( \text{Pop}_{\text{iniBest}} \)
10. calculate the fitness of all solutions in \( \text{Pop}_{\text{iniBest}} \)
11. find the best solution referred as \( \text{Sol}_{\text{best}} \)
12. --- Stage 2: Improvement: Salp Swarm Algorithm ---
13. as in Algorithm 1 (from line-6 to line-30)

**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 SSA_GA-tuner. In SSA_GA-tuner, GA plays a crucial role in determining the optimal parameters for SSA.

**Initialization** Initially the individual of GA is represented as a vector (i.e., \( y = (p_1, p_2) \)) 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 benchmark function with predefined population size and specific maximum number of iteration (i.e. max Itr). The results obtained by SSA for each individual is considered as the fitness function value. For GA, the initial population of size \( (GA_{\text{popSize}}) \) 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 \( (GA_{\text{pop}}) \). Formally, let the \( \varphi_i \) is the selection probability of the individual \( i \). The value of \( \varphi_i \) is calculated as in Eq. (8).

\[
\varphi_i = \frac{f(x_i)}{\sum_{j=1}^{\text{PopSize}} f(x_j)}
\]

Note that the \( \sum_{j=1}^{\text{PopSize}} \varphi_i \) is unity. For example, Fig. 9 pie-charts the probability of five individual \( GA_{\text{pop}} = (x_1, x_2, x_3, x_4, x_5) \) where the fitness vector is \( f(x_1) = 0.54, f(x_2) = 1.5, f(x_3) = 2.66, f(x_4) = 3.22, f(x_5) = 4.13 \). 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.

**Encoding** In the encoding step, the whole decision variables in the individuals stored in \( GA_{\text{pop}} \) are reformulated using a binary format. For example, let the individual be \( x = [6, 14] \), it will be reformulated to binary as follows: \( x = [0110, 1110] \). It is worth noting that, the individual is
called a chromosome, and each bit of the chromosome is called a gene. Finally, Fig. 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 Fig. 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 $\gamma_r \in [0, 1]$ is used to determine the probability of using a crossover operator. The higher value of $\gamma_r$ close 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 is not primitive random. An example of this operator is shown in Fig. 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 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 Sect. 4.1 is used in SSA to generate the initial population.
Algorithm 4 Evaluating GA Individuals \((p_1, p_2)\) using SSA

Pseudo Code

1: set GA population size referred as \(GA\text{PopSize}\)
2: set No of SSA Runs referred as \(repetition\)
3: set counter \(j\) \(+\)
4: while \((i \leq GA\text{PopSize})\) do
5:   set counter \(k\) \(+\)
6:   while \((k \leq repetition)\) do
7:     Calculate \((SSA(f_j, Sol_i))\) using \(Pop_{init}\)
8:     Calculate \((best - average(f_j, Sol_i))\)
9:     \(k + +\)
10: end while
11: set counter \(i\) \(+\)
12: while \((i \leq GA\text{PopSize})\) do
13:   \(sum(Sol_i) = sum(Sol_i) + best - average(f_j - Sol_i)\)
14:   \(f(Sol_i) = sum(Sol_i)/repetition\)
15: \(i + +\)
16: end while
17: \(j + +\)
18: 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.

\[
c_1 = p_1e^{-\left(\frac{x}{c}\right)^2} \tag{9}
\]

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 \(SSA_{std}\) 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 \(SSA_{GA-tuner}\) is studied and compared against a diversified population \(SSA_{std}\) 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.

5.1 Benchmark functions

The benchmark functions are grouped into two types, a unimodal and multi-modal, and are listed with their mathematical formulations, boundaries, global optima, and dimension in the Tables 4 and 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 Mirjalili et al. (2017), 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.
Table 3 Parameters settings

| Parameter                                      | Standard SSA | SSA\text{std} | SSA\text{GA−tuner} |
|------------------------------------------------|--------------|----------------|--------------------|
| SSA max iteration \((L)\)                     | 500          | 500            | 500                |
| SSA termination criteria \((L)\)              | 500          | 500            | 500                |
| SSA population size \((N)\)                   | 20           | 20             | 20                 |
| SSA decision variables No.                    | Based on -   | Based on -     | Based on -         |
| Benchmark function                            |              | benchmark function | benchmark function |
| No. of Runs                                    | 31           | 31             | 31                 |
| Max No. of initial population \((\text{Max\#Pop})\) | –            | 2000          | 2000               |
| GA maximum generation \((\text{GA}\text{MaxGen})\) | –            | –             | 2000               |
| GA termination criteria \((\text{GA}\text{MaxGen})\) | –            | –             | random             |
| GA population size \((\text{GA}\text{PopSize})\) | –            | –             | 31                 |
| \(p_1\) and \(p_2\)                         | –            | –             | Determined by GA   |
| Calculated by Eq. (3)                         | Calculated by Eq. (3) | Calculated by Eq. (9) |
| \(c_1\) and \(c_2\)                         | Random       | Random         | Random             |
| \((lb)\) and \((ub)\)                       | Based on -   | Based on -     | Based on -         |
| Benchmark function                            | Benchmark function | Benchmark function | 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        |

5.2 Effect of diversified population on SSA\text{std}

The comparison among proposed diversified SSA\text{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 SSA\text{std} algorithm is able to obtain the best results and outperforms the standard SSA in almost all tested functions. Results of SSA\text{std} algorithm demonstrate that the more diverse initial population has a remarkable positive impact on the quality of the algorithm’s final results.

5.3 The effect of self-adaptive parameter tuning on SSA\text{GA−tuner}

The comparison between proposed tuned SSA\text{GA−tuner} algorithm, diversified SSA\text{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 SSA\text{GA−tuner} outperforms both SSA\text{std} and the standard SSA for almost all tested functions. In addition, SSA\text{GA−tuner} is able to obtain the best result in all functions in comparison with the other two algorithms. Results of SSA\text{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 SSA algorithm has significant difference (p-value < 0.05) in comparison...
Table 7  p-values of Wilcoxon test for Standard SSA, SSAstd, and SSA\textsubscript{GA−tuner} Algorithms: best obtained results for employed benchmark functions

| No. | Function | SSA\textsubscript{std} vs. standard SSA | SSA\textsubscript{GA−tuner} vs. standard SSA | SSA\textsubscript{GA−tuner} vs. SSA\textsubscript{std} |
|-----|-----------|----------------------------------------|---------------------------------------------|---------------------------------------------|
| F1  | Sphere    | 0.000001                               | 0.001197                                    | 0.000001                                    |
| F2  | Rastrigin | 0.377861                               |                                             |                                             |
| 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

Table 8  Computational time

| No. | Function | Standard SSA | SSA\textsubscript{std} | SSA\textsubscript{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               | 41.855096                  |
| 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

with Standard SSA algorithm except on functions: F2, F4, F6, F9, and F10 and comparing with SSA\textsubscript{GA−tuner} (except on functions: F4, F6, F7, F8, F9, F11, and F12). On the other hand, obtained results for SSA\textsubscript{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 SSA\textsubscript{std} and SSA\textsubscript{GA−tuner} results, as there are seven functions with p-values greater than 0.05.

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 SSA\textsubscript{std} algorithm, search for high diverse population before moving to the optimization process, and the SSA\textsubscript{GA−tuner} algorithm has two additional tasks in addition to the main optimization process (i.e., searching for a high diverse population and finding the optimal parameter values for a specific
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 |

| Function       | Statistical measure | 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 | 1.40E-02 | 1.70E-01 | 1.18E-02 | 1.44E-01 |
| Rosenbrock     | Avg                 | 6.95E+00 | 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 |
| Michalewicz    | 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

problem). In addition, it is noticeable that the 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.

5.5 Comparative evaluation

To validate our work, two comparisons with state of the art methods were conducted. These state-of-the-art meth-
ods 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 Harifi et al. (2019) that were adopted in this research. Note that the results of seven benchmark functions in the second comparison used ten benchmark functions in the first comparison. 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\textsubscript{GA–tuner} |
|-----------|---------------------|-------------|-------------|--------------|--------------|-------------|--------------|--------------|--------------------------|
| Sphere    | Avg                 | 2.82E+02    | 7.58E-14    | 9.75E-01     | 3.33E-12     | 2.76E-13    | \textbf{0.0E+00} | \textbf{0.0E+00} | 1.23E-44            |
|           | std                 | 3.16E+01    | 1.08E-13    | 1.97E-01     | 1.03E-12     | 9.44E-14    | \textbf{0.0E+00} | \textbf{0.0E+00} | 6.72E-44            |
| Rastrigin | Avg                 | 1.44E+02    | 1.90E+02    | 1.87E+02     | 1.79E+01     | 1.87E+02    | 1.17E+01     | 2.01E+01     | 1.19E+00            |
|           | 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     | 1.11E+00            |
|           | std                 | 4.67E-02    | 4.79E-02    | 5.29E-02     | 5.62E-02     | 1.59E-01    | 7.14E-02     | 4.54E-02     | 1.14E-01            |
| 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                 | \textbf{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.

SSA\textsubscript{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 SSA\textsubscript{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 SSA\textsubscript{GA–tuner} algorithm performance is stable.

For the second comparison, it can be seen that the proposed SSA\textsubscript{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 get 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 SSA\textsubscript{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 SSA\textsubscript{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 SSA\textsubscript{std} version is studied. The pro-
posed 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 SSA_{std}. In order to evaluate the impact of the self-adaptive parameter control on the convergence of SSA_{GA-tuner}, the comparative results against standard SSA and SSA_{std} show that the 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.

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

Funding The authors have not disclosed any funding.

Data Availability Statement Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest 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

Abbassi R, Abbassi A, Heidari AA, Mirjalili S (2019) An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. Energy Convers Manage 179:362–372

Abdullah S, Nceef SK, Turky A (2018) An interleaved artificial bee colony algorithm for dynamic optimisation problems. Connect Sci 30(3):272–284

Ahn CW (2006) Practical genetic algorithms, Advances in Evolutionary Algorithms: Theory, Design and Practice 7–22

Al Nsour H, Alweshah M, Hammouri AI, Al Ofeshat H, Mirjalili S (2018) A hybrid grey wolf optimiser algorithm for solving time series classification problems. J Intel Syst 29(1):846–857

Alresheedi SS, Lu S, Abd Elaziz M, Ewees AA (2019) Improved multi-objective salp swarm optimization for virtual machine placement in cloud computing. Human-centric Comp Inform Sci 9(1):15

Altan A, Karasu S (2020) 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 140:1100711071

Alweshah M (2014) Firefly algorithm with artificial neural network for time series problems. Res J Appl Sci Eng Technol 7(19):3978–3982

Alweshah M (2018) Construction biogeography-based optimization algorithm for solving classification problems. Neural Comput Appli 31(10):5679–88

Alweshah M, Abdullah S (2015) Hybridizing firefly algorithms with a probabilistic neural network for solving classification problems. Appl Soft Comput 35:513–524

Alweshah M, Hammouri AI, Tedmori S (2017) Biogeography-based optimisation for data classification problems. Int J Data Mining, Modell Manag 9(2):142–162

Asithambhi S, Rajappa M (2018) Swarm intelligence-based approach for optimal design of cmos differential amplifier and comparator circuit using a hybrid salp swarm algorithm. Rev Sci Instrum 89(5):054702

Atashpaz-Gargari E, Lucas C, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: (2007) IEEE congress on evolutionary computation. IEEE 2007:4661–4667

Ateya AA, Muthanna A, Vybornova A, Algarni AD, Abuarqoub A, Koucheryavy Y, Koucheryavy A (2019) Chaotic salp swarm algorithm for sdn multi-controller networks. Eng Sci Technol Inter J 22(4):1001–1012

Balande U, Shirimankar D (2019) Srína: Stochastic ranking with improved-firefly-algorithm for constrained optimization engineering design problems. Mathematics 7(3):250

Barik AK, Das DC, 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 2018:1–6

Bernstein G, O’Brien K (2013) Stochastic agent-based simulations of social networks, In: Proceedings of the 46th annual simulation symposium, Society for Computer Simulation International, p. 5

Blackwell TM, Bentley P (2002) Don’t push me! collision-avoiding swarms, In: Proceedings of the 2002 congress on evolutionary computation. CEC’02 (Cat. No. 02TH8600), Vol. 2, IEEE, pp. 1691–1696

Blanquart F (2019) Evolutionary epidemiology models to predict the dynamics of antibiotic resistance. Evol Appl 12(3):365–383

Brits R, Engelbrecht AP, Van den Bergh F (2002) A niching particle swarm optimizer, In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, Vol. 2, Singapore: Orchid Country Club, pp. 692–696
Self-adaptive salp swarm algorithm...
ing and novel local search algorithm for feature selection. Expert Syst Appl 145:113122
Turky A, Abdullah S, Dawod A (2018) A dual-population multioperators harmony search algorithm for dynamic optimization problems. Comput Indus Eng 117:19–28
Wang J, Gao Y, Chen X (2018) 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):1561
Wilhelm PA (2008) Pheromone particle swarm optimization of stochastic systems, Ph.D. thesis
Wolpert DH, Macready WG et al (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
Wu J, Nan R, Chen L (2019) Improved salp swarm algorithm based on weight factor and adaptive mutation. J Exp Theor Artif Intell 31(3):493–515
Xie X-F, Zhang W-J, Yang Z-L (2002) Dissipative particle swarm optimization, in: Proceedings of the 2002 congress on evolutionary computation. CEC’02 (Cat. No. 02TH8600), Vol. 2, IEEE, pp. 1456–1461
Yin Y, Tu Q, Chen X (2020) Enhanced salp swarm algorithm based on random walk and its application to training feedforward neural networks. Soft Comput 24(19):14791–807
Yu W-J, Shen M, Chen W-N, Zhan Z-H, Gong Y-J, Lin Y, Liu O, Zhang J (2014) Differential evolution with two-level parameter adaptation. IEEE Trans Cybern 44(7):1080–1099
Yu Y, Wang H, Li N, Zhang H, Su Z, Shao X (2018) Finite-time model-assisted active disturbance rejection control with a novel parameters optimizer for hypersonic reentry vehicle subject to multiple disturbances. Aerosp Sci Technol 79:588–600
Zandevakili H, Rashedi E, Mahani A (2019) Gravitational search algorithm with both attractive and repulsive forces. Soft Comput 23(3):783–825
Zhang, Yu Y, Zheng S, Todo Y, Gao S (2018) Exploitation enhanced sine cosine algorithm with compromised population diversity for optimization, In: 2018 IEEE International conference on progress in informatics and computing (PIC), IEEE, pp. 1–7
Zhang J, Wang Z, Luo X (2018) Parameter estimation for soil water retention curve using the salp swarm algorithm. Water 10(6):815
Zhang L, Li C, Wu Y, Huang J, Cui Z (2018) An improved salp swarm algorithm with spiral flight search for optimizing hybrid active power filters’ parameters. IEEE Access 8:154816–154832
Zhang Q, Chen H, Heidari AA, Zhao X, Xu Y, Wang P, Li Y, Li C (2019) Chaos-induced and mutation-driven schemes boosting salp chains-inspired optimizers, IEEE. Access 7:31243–31261
Zheng H, Cai Z, Ye X, Wang M, Kuang F, Chen H, Li C, Li Y (2020) A multi-strategy enhanced salp swarm algorithm for global optimization. Eng Comput 10:1–27
Zhang W, Hou W, Yang D, Xing Z, Gen M (2019) Multiobjective pso algorithm with multi-directional convergence strategy to solve flow shop scheduling problems. In: International conference on management science and engineering management, Springer, pp. 750–759
Zhang J, Wang J Improved salp swarm algorithm based on levy flight and sine cosine operator, IEEE Access
Zhao T-H, Khan ML, Chu Y-M (2021) Artificial neural networking (ANN) analysis for heat and entropy generation in flow of non-newtonian fluid between two rotating disks, Math Methods Appl Sci
Zhao X, Zhang X, Cai Z, Tian X, Wang X, Huang Y, Chen H, Hu L (2019) Chaos enhanced grey wolf optimization wrapped elm for diagnosis of paraquat-poisoned patients. Comput Biol Chem 78:481–490
Zhao X, Yang F, Han Y, Cui Y (2020) An opposition-based chaotic salp swarm algorithm for global optimization. IEEE Access 8:36485–36501
Zhu W, Tang Y, Fang J-A, Zhang W (2013) Adaptive population tuning scheme for differential evolution. Inf Sci 223:164–191
Zivkovic M, Stoean C, Chhabra A, Budimirovic N, Petrovic A, Bacanin N (2022) Novel improved salp swarm algorithm: an application for feature selection. Sensors 22(5):1711

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.