Advances in Engineering Software 114 (2017) 163–191

Contents lists available at ScienceDirect

Advances in Engineering Software

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

Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems

Seyedali Mirjalili[a,*], Amir H. Gandomi[b,f], Seyedeh Zahra Mirjalili[c], Shahrzad Saremi[a], Hossam Faris[d], Seyed Mohammad Mirjalili[e]

[a] Institute for Integrated and Intelligent Systems, Griffith University, Nathan, QLD 4111, Australia
[b] School of Business, Stevens Institute of Technology, Hoboken, NJ 07030, USA
[c] School of Electrical Engineering and Computing, University of Newcastle, Callaghan, NSW 2308, Australia
[d] Business Information Technology Department, King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
[e] Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, H3G1M8, Canada
[f] BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 488241, USA

ARTICLE INFO

Article history:
Received 25 December 2016
Revised 23 April 2017
Accepted 9 July 2017
Available online 24 July 2017

Keywords:
Particle swarm optimization
Multi-objective optimization
Genetic algorithm
Heuristic algorithm
Algorithm
Benchmark

ABSTRACT

This work proposes two novel optimization algorithms called Salp Swarm Algorithm (SSA) and Multi-objective Salp Swarm Algorithm (MSSA) for solving optimization problems with single and multiple objectives. The main inspiration of SSA and MSSA is the swarming behaviour of salps when navigating and foraging in oceans. These two algorithms are tested on several mathematical optimization functions to observe and confirm their effective behaviours in finding the optimal solutions for optimization problems. The results on the mathematical functions show that the SSA algorithm is able to improve the initial random solutions effectively and converge towards the optimum. The results of MSSA show that this algorithm can approximate Pareto optimal solutions with high convergence and coverage. The paper also considers solving several challenging and computationally expensive engineering design problems (e.g. airfoil design and marine propeller design) using SSA and MSSA. The results of the real case studies demonstrate the merits of the algorithms proposed in solving real-world problems with difficult and unknown search spaces.

1. Introduction

Over the past decade, meta-heuristic techniques have become surprisingly very popular. This popularity is due to several main reasons: flexibility, gradient-free mechanism, and local optima avoidance of these algorithms. The first two advantages originate from the fact that meta-heuristics consider and solve optimization problems by only looking at the inputs and outputs. In other words, meta-heuristics assume an optimization problem as a black box. Therefore, there is no need to calculate derivative of the search space. This makes them highly flexible for solving a diverse range of problems. Since meta-heuristics belong to the family of stochastic optimization techniques, they benefit from random operators. This assists them to avoid local solutions when solving real problems, which usually have a large number of local optima. Due to these advantages, the application of meta-heuristics can be found in different branches of science and industry.

Meta-heuristic algorithms are classified into two dominant classes: evolutionary [1] and swarm intelligence [2] techniques. Evolutionary algorithms mimic the concepts of evolution in nature. The best and most well-regarded algorithm in this class is Genetic Algorithm (GA) [3]. This algorithm simulates the concepts of Darwinian theory of evolution. In GA, the optimization is initiated with a set of random solutions for a particular problem. After evaluating the solutions by the objective function, it modifies the variables of solutions based on their fitness value. Since the best individuals are given higher probability to involve in improving other solutions, the random initial solutions are very likely to be improved. There are several other evolutionary algorithms in the literature such as Differential Evolution (DE) [4], Evolutionary Strategy (ES) [5], and Evolutionary Programming (EP) [6,7], and Biogeography-Based Optimization (BBO) algorithm [8] as well.

Swarm intelligence techniques mimic the intelligence of swarms, herds, schools, or flocks of creatures in nature. The main foundation of these algorithms originates from the collective behaviour of a group of creatures. For instance, ants are able to collectively guarantee the survival of a colony without having a centralized control unit. In other word, no one tells ants where and

* Corresponding author.
E-mail address: seyedali.mirjalili@griffithuni.edu.au (S. Mirjalili).
URL: http://www.alimirjalili.com/ (S. Mirjalili)

http://dx.doi.org/10.1016/j.advengsoft.2017.07.002
0965-9978/

how a source food can be found, but they cooperatively find foods at even far distances from their nests. The two most popular algorithms in this class are Ant Colony Optimization (ACO) [9] and Particle Swarm Optimization (PSO) [10]. The ACO algorithm mimics the social behaviour of ants for finding the shortest path between the nest and a source food. The PSO algorithm simulates the collective behaviour of birds in navigating and hunting. Other swarm intelligence techniques in the literature are: Artificial Bee Colony (ABC) algorithm [11], Cuckoo Search (CS) algorithm [12], Firefly Algorithm (FA) [13], Bat Algorithm (BA) [14], Grey Wolf Optimizer (GWO) [15–17], Dolphin Echolocation (DE) [18], Whale Optimization Algorithm (WOA) [19], Fruitfly Optimization Algorithm (FOA) [20], and Harmony Search [21,22].

Regardless of the difference between evolutionary and swarm intelligence techniques, the common is the improvement of one or a set of solutions during optimization. If an algorithm improves only one solution, it is called individualist algorithm. If a set of solutions is improved, it is referred as a collective algorithm. Individualist algorithms are beneficial because of the low number of required function evaluation and simplicity of the overall optimization process. However, the probability of local optima stagnation is very high. Collective algorithms are able to avoid local solutions better and exchange information about the search space. However, such techniques require more number of function evaluations. Some of the individualist algorithms are Tabu Search (TS) [6,23], hill climbing [24], Iterated Local Search (ILS) [25], and Simulated Annealing (SA) [26], Variable Neighborhood Search (VNS) [27], and Guided Local Search [28]. The well-known collective algorithms are GA, ACO, PSO, DE, and ES.

Despite the merits of the proposed algorithms in the literature, it has been proved by the No-Free-Lunch (NFL) [29] that none of these algorithms are able to solve all optimization problems. In other words, all meta-heuristics perform similar when solving all optimization problems. This theorem reveals the importance of new and specific algorithms in different fields because effectiveness of an algorithm in solving a set of problems does not guarantee its success in different sets of test problems. This is the motivation of this paper, in which a new meta-heuristic optimization algorithm is first proposed for solving single-objective problems and then extended to a multi-objective version. The rest of the paper is organized as follows.

Section 2 reviews the literature and relevant works. Section 3 presents the inspiration and mathematical model proposed. The Salp Swarm Algorithm (SSA) and Multi-objective Salp Swarm Algorithm (MSSA) are proposed in this section as well. The qualitative and quantitative results of both algorithms on a variety of benchmark functions are presented and discussed in Section 4. Both SSA and MSSA are employed to solve several challenging real problems in Section 5. Finally, Section 6 concludes the work and suggest several future research directions.

2. Related works

This section reviews the state-of-the-art in the field of stochastic optimization. There are many branches in this field such as single-objective, multi-objective, constrained, dynamic, surrogate-assisted, many-objective, and so on. Since the algorithms proposed solve single- and multi-objective optimization problems, the main focus of this section is on the challenges and related works in single- and multi- objective optimization fields.

2.1. Single-objective optimization problems

As its name implies, single-objective optimization deals with one objective. This means there is only one objective to be minimized or maximized. This type of optimization might be subject to a set of constraints as well. The constraints are divided to two categories: equality and inequality. Single-objective optimization is formulated as a minimization problem as follows (without the loss of generality):

$$Minimize: \ F(\vec{x}) = \{f_1(\vec{x})\} \tag{2.1}$$

$$Subject\ to: \ g_i(\vec{x}) \geq 0, \ i = 1, 2, \ldots, m \tag{2.2}$$

$$h_i(\vec{x}) = \ 0, \ i = 1, 2, \ldots, p \tag{2.3}$$

$$lb_i \leq x_i \leq ub_i, \ i = 1, 2, \ldots, d \tag{2.4}$$

where d is the number of variables, p is the number of equality constraints, m is the number of inequality constrained, lb_i is the lower bound of the ith variable, and ub_i indicates the upper bound of the ith variable.

The set of variables, objectives, range of variables, and constraints create a search space/landscape. This search space exists in a d-dimensional space where d is the number of variables. For 1D, 2D, and 3D problems, we can easily draw the search space in a Cartesian coordinate system and observe their shapes. However, it is not possible to draw dimensions greater than 3 because they are beyond the dimensions that we experience every day. Therefore, a large number of variables is the first challenge when solving optimization problems.

The range of variables confides the search space and is varied. The variables themselves can be continuous or discrete, in which they create either a continuous or a discrete search space. In a former case, there is an infinite number of points between each two points in the search space. In the latter case, however, there is a finite set of points between two points. Finding the global optimum in a continuous space is different from a discrete one, and each of them has their own challenges. Although most of the optimization problems come with range of variables, there are some problems that do not have a specific range to be considered during optimization. An example is the problem of training Neural Networks (NNs) [30]. The connection weights and biases can be any real number. Solving such problems also need special consideration. For instance, an optimizer might start with an initial range and then expand it during optimization.

The constraints limit the search space even further. They create gaps in the search space because the solutions in those regions are not suitable for the problem. For instance, the thickness of a propeller blade cannot go below a certain number due to the fragility. A set of constraints can even split the search space to different separated regions. The solutions that violate the constrained regions are called infeasible solutions. In contrast, the solutions inside the constrained areas are called feasible solutions. In the literature, there are two terms for the parts of the search space that are inside and outside the constrained areas: feasible and infeasible regions. A constrained search space has the potential to make an algorithm ineffective despite its good performance in an unconstrained search space. Some of the real problems such as Computational Fluid Dynamic problems have dominated infeasible regions. Therefore, optimization techniques should be equipped with suitable operators [31] to handle constraints as well.

Another challenge when solving optimization problems is the presence of local solutions. The search space that the variables, objective function, and constraints create may be very simple or complicated. In most of the works in the literature, the number of local solutions is considered as the main difficulty for optimization algorithms. In a single-objective search space there is one best solution (the so-called global optimum) that returns the best objective value. However, there are usually many other solutions that return values close the objective value of the global optimum. This kind of solutions are called local solutions because they are locally the best solution if we consider the search space in their vicinity, but they are not the best solution globally when considering

the entire search space. The presence of local solutions causes the entrapment of many optimization algorithms. Local optima stagnation refers to the situation where an optimization algorithm finds a local solution and mistakenly assumes it as the global optimum. A real search space usually has a large number of local solutions. Therefore, an optimization algorithm should be able to avoid them efficiently to determine the global optimum.

The convergence speed is also a difficulty when solving optimization problems. An algorithm that is able to avoid local solutions is not necessarily able to converge towards the global optimum. Finding the rough location of the global optimum is done when an algorithm avoids local solutions. The next step is to improve the accuracy of the rough solutions obtained. The term convergence in the literature refers to the rate or behaviour of an algorithm towards the global optimum. Of course, a quick convergence results in local optima stagnation. By contrast, sudden changes in the solutions lead to local optima avoidance but reduce the convergence speed towards the global optimum. These two trade-offs are the main challenges that an algorithm deals with when solving real problems. Converge speed is essential in finding an accurate approximation of the global optimum.

There are other types of difficulties when solving real single-objective search spaces such as: deceptive optimum, isolation of the optimum, uncertainty, noisy objective function, dynamic objective function, reliable optimum, and so on. Each of these difficulties needs special consideration. These concepts are out of the scope of this work, so interested readers are referred to the surveys conducted by Boussaid [32].

2.2. Multi-objective optimization problems

As its name implies, multi-objective optimization deals with more than one objective. All of the objectives should be optimized simultaneously to solve multi-objective problems. Multi-objective optimization is formulated as follows (without the loss of generality):

$$Minimize: \ F(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_o(\vec{x})\} \tag{2.5}$$

$$Subject\ to: \ g_i(\vec{x}) \geq 0, \ i = 1, 2, \ldots, m \tag{2.6}$$

$$h_i(\vec{x}) = \ 0, \ i = 1, 2, \ldots, p \tag{2.7}$$

$$lb_i \leq x_i \leq ub_i, \ i = 1, 2, \ldots, n \tag{2.8}$$

where o is the number of objectives, m is the number of inequality constraints, p is the number of equality constraints, lb_i is the lower bound of the ith variable, and ub_i indicates the upper bound of the ith variable.

The nature of such problems prevents us from comparing solutions using the relational operators. This is because there is more than one criterion to compare solutions. With one objective we can confidently say if a solution is better than another using relational operators, but with more than one objective we need other operator(s). The main operator to compare two solutions considering multiple objectives is called Pareto optimal dominance and defined as follows [33]:

Definition 1. Pareto domination:

Assuming the following two vectors: $\vec{x} = (x_1, x_2, \ldots, x_k)$ and $\vec{y} = (y_1, y_2, \ldots, y_k)$.

Vector x dominates vector y ($x \prec y$) if and only if:

$$\forall i \in \{1, 2, \ldots, k\}, \ \left[f_i\left(\vec{x}\right) \leq f_i\left(\vec{y}\right)\right] \ \wedge \ \exists i \in \{1, 2, \ldots, o\}, \ \left[f_i\left(\vec{x}\right) < f_i\left(\vec{y}\right)\right] \tag{2.9}$$

Inspecting Eq. (2.9), it may be seen that a solution is better than another if it has equal and at least one better value in the objectives. In this case, it is said that a solution dominates the other. If this does not hold for two solutions, they are called Pareto optimal or non-dominated solutions. The answers for a multi-objective problem are the non-dominated solutions. The Pareto optimality is also important in multi-objective optimization and defined as follows [34]:

Definition 2. Pareto optimality:

Assuming that $\vec{x} \in X$, $\vec{x}$ is a Pareto-optimal solution if and only if:

$$\left\{\nexists \ \vec{y} \in X | \ \ \vec{y} \prec \vec{x}\right\} \tag{2.10}$$

For every multi-objective problem, there is a set of Pareto optimal solutions, which represents the best trade-offs between the multiple objectives. In optimization, this solution set is called Pareto optimal set. The projection of the Pareto optimal solutions in the objective space is called Pareto optimal front. These two sets are defined as follows:

Definition 3. Pareto optimal set:

The Pareto set is a set that includes the Pareto optimal solutions:

$$P_s := \left\{\vec{x}, \ \ \vec{y} \in X | \ \nexists \ \vec{y} \prec \vec{x}\right\} \tag{2.11}$$

Definition 4. Pareto optimal front:

This set is consisted of the objective values for the solutions in the Pareto solutions set:

$$\forall i \ \in \{1, 2, ..., o\}, \ P_f := \left\{f_i\left(\vec{x}\right) | \ \vec{x} \in P_s\right\} \tag{2.12}$$

With these four definitions, solutions can be easily compared to solve multi-objective problems. The set of variables, objectives, and constraints again create a search landscape. The main difference between a multi-objective search space and a single-objective one is that there are multiple objectives. Therefore, illustration of the search space is difficult for problems with more than three objectives. This is why researchers normally consider two search spaces: parameter space and objective space.

Similarly to single-objective search spaces, the range of variables determine the boundaries of the search space in each dimension and constrains confide them. The effects of equality and inequality constraints on a multi-objective search space are very similar to those in a single-objective search space. For every multi-objective problem, there is a set of best trade-offs between objectives which is called true Pareto optimal front. There are three main challenges for a multi-objective optimization technique to find a Pareto optimal front: local fronts, convergence, and distribution of solutions (coverage).

Local solutions and slow convergence are the common issues between single- and multi-objective optimization fields. Due to the presence of multiple solutions in the Pareto optimal front, however, the distribution of solutions is also important in multi-objective optimization. The ultimate goal is to find a uniformly distributed front to give decision makers a lot of options for decision making. Multi-objective problems have fronts with different shapes: concave, convex, linear, separated, etc. Finding a well-distributed Pareto optimal front for each of these shapes is very challenging and should be addressed well in *a posteriori* methods. Such methods are discussed in detail in the Section 2.4.

There are other types of difficulties when solving real multi-objective problems: deceptive front, isolation of the front, uncertainty, noisy objective functions, dynamic objective functions reliable fronts, and so on. Each of these difficulties need special consideration. These concepts are out of the scope of this work, so

interested readers are referred to the surveys conducted by Zhou et al. [35].

2.3. Single-objective optimization algorithms

The optimization algorithms in the literature can be divided into two main classes: deterministic versus stochastic. Deterministic algorithms always find the same solution for a given problem if they start with the same starting point. The main advantage of such methods is reliability because they definitely find a solution in each run. However, local optima stagnation is a drawback since these algorithms usually do not have randomized behaviours when solving optimization problems. The second class of optimization algorithms, stochastic methods, benefits from stochastic operators. This results in finding different solutions even if the starting point remains unchanged and consequently makes stochastic algorithms less reliable compared to the deterministic approaches. However, the randomized behaviour promotes local optima avoidance, which is the main advantage of stochastic algorithms. The reliability of the stochastic methods can be improved by fine tuning and increasing the number of runs.

Stochastic optimization algorithms are divided into two categories: individualist and collective. In the first family, a stochastic algorithm starts and perform optimization with a single solution. This solution is randomly changed and improved for a pre-defined number of steps or satisfaction of an end criterion. The most well-regarded algorithms in this family are TS [6,23], hill climbing [24], ILS [25], and SA [26]. The advantage of this group is the low computational cost and the need for a low number of function evaluations.

However, collective techniques create multiple random solutions and evolve them during optimization. The collection of solutions usually collaborates to better determine the global optimum in the search space. The most notable algorithms in this field are: GA, PSO, ACO, and DE. Multiple solutions decrease the chance of local optima stagnation, which is the main advantage of collective algorithms. However, each of the solutions need one function evaluation and establishing effective collaborations between the solutions is challenging. Despite these two drawbacks, collective stochastic optimization techniques are very popular these days. The application of such methods can be found widely in science and industry. One of the main reason is the high capability in avoiding local solutions.

As mentioned above, real search spaces have a significantly large number of local solutions. Therefore, collection of solutions and random components both contribute in finding a better solution for real problems. Another advantage of collective algorithms is flexibility. Such techniques are able to collaboratively overcome the challenges in different types of search spaces. There is also no need for gradient information because stochastic collective algorithms consider a problem as a black box and only monitor the inputs, outputs, and constraints of the problem. Last but not least, simplicity is the other advantage because most of the collective algorithms mimic simple rules in swarms, flocks, schools, or packs.

Regardless of the differences between collective stochastic algorithms, they all follow the same procedure to approximate the global optimum. The optimization first starts by a set of random solutions, and they are required to combine and change randomly, rapidly, and abruptly. This causes that the solutions move globally. This process is called exploration of the search space because the solutions are gravitated towards different regions of the search space by sudden changes. The primary goal in the exploration phase is to determine promising areas of the search space and to avoid local solutions. After enough exploration, the solutions start changing gradually and move locally. This process is called exploitation where the main goal is to improve the accuracy of the best solutions obtained in the exploration phase. Although local optima avoidance may occur in the exploitation phase, the coverage of search space is not as broad as the exploration phase. In this case, the solutions avoid local solutions in the vicinity of the global optimum.

The above paragraph shows that the exploration and exploitation phases seek conflicting objectives. The question here is as to when the best time is to transit from exploration to exploitation. Due to the stochastic nature of collective algorithms and unknown shape of the search space, no one is able to answer this question. Therefore, most of the algorithms smoothly require the search agents to move from exploration phase to exploitation using adaptive mechanisms [36].

Due to the advantages of stochastic collective algorithms, there is an increasing interest in proposing new algorithms in this field. They can be divided to four main classes based on inspiration: swarm-inspired, physics-inspired, evolutionary, and human-based. The most popular swarm inspired algorithms are PSO, ACO, ABC [11] optimization, CS [12], FA [13], GWO [15], and Krill Herd (KH) [37–39] algorithm. The most popular physics-based ones are Gravitational Search Algorithm (GSA) [40], Charged System Search (CSS) [41], Central Force Optimization (CFO) [42], and Ray Optimization (RO) [43]. The most popular evolutionary algorithms are BBO [8], ES [5], GA, and DE [4]. Finally, the most well-known human-inspired group are Teaching and Learning Based Optimization algorithms (TLBO) [44], Seeker Optimization Algorithm (SOA) [45], Soccer League Competition (SLC) algorithm [46], and Mine Blast Algorithm (MBA) [47].

2.4. Multi-objective optimization algorithms

There are two main approaches for solving multi-objective problems: *a priori* versus *a posteriori* [48]. In the former method, the multi-objective problem is converted to a single-objective one by aggregating the objectives. This is done using a set of weights that is usually defined by an expert and dictates the importance of each objective. After aggregation of the objectives, a single-objective optimizer can be employed to find the optimal solution. The main drawback of this technique is that the Pareto optimal set and front can be constructed by re-running the algorithm and changing the weights. In addition, the concave regions of the Pareto optimal front cannot be determined due to the nature of additive weights in such methods [49–51]. There are some improvements in the literature for the former issue but all of them still need to be run multiple times to find more than one Pareto optimal solution [52].

However, *a posteriori* approach maintains the multi-objective formulation of the problem. The main advantage is that the Pareto optimal set can be determined in a single run [53]. Also, any type of Pareto optimal front can be approximated and there is no weight to be defined by experts. With this approach, there would be many different solutions for decision maker compared to *a priori* methods. On the other hand, the main drawback is the need for addressing multi-objectives and special mechanisms to determine the Pareto optimal set and front. This make *a posteriori* optimization more challenging and computationally more expensive. Since the advantages of *a posteriori* optimization are more than its drawbacks compared to *a priori* methods, the focus of this work is on *a posteriori* multi-objective optimization.

Multi-objective optimization techniques in the literature of stochastic optimization have been equipped with new mechanisms to find an accurate approximation of the true Pareto optimal solutions and front with a uniform distribution. One of the most popular algorithms in this field is Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [54]. This algorithm utilizes an aggregation method to create sub-problems from a given

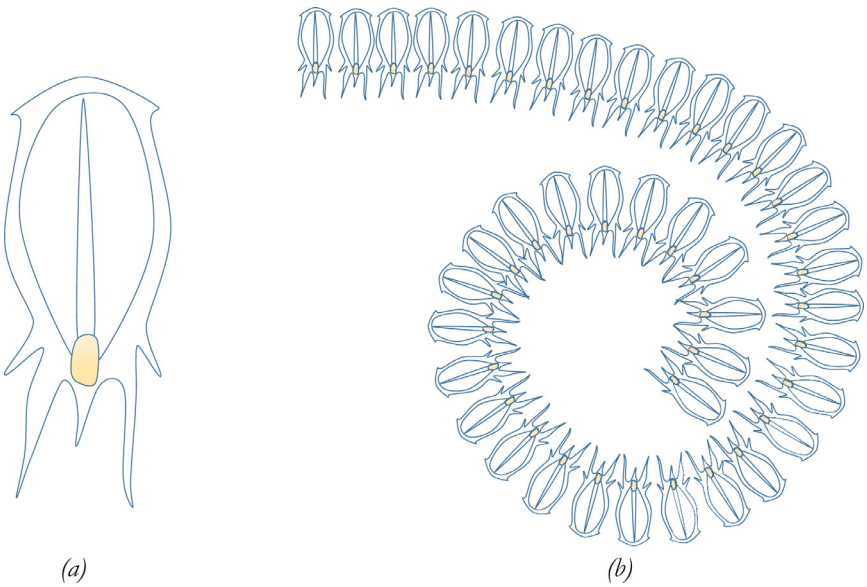

Fig. 1. (a) Individual salp, (b) swarm of salps (salps chain).

problem. Each sub-problem is a single-objective problem but the main difference between MOEA/D and a pure aggregation method is that sub-problems are optimized by using information from other neighbouring sub-problems. In other words MOEA/D uses evolutionary operators to combine and evolve sub-problems over the course of generations. This algorithm has several improvements and variants that can be found in [35].

DE-based methods [55,56] have been also popular in the literature. For designing the multi-objective version of DE, the reproduction operators have been modified widely. For instance in Pareto-frontier Differential Evolution (PDE) [57,58] the parents are selected from the set of non-dominated solutions to perform cross over. In PSO-based techniques, finding non-dominated solutions and updating non-dominated solutions are easy but choosing one of them as the best solution (*gbest* or *pbest* in PSO for instance) and storing the most distributed ones are challenging [59,60]. Many works have been done recently in this area. Popular approaches are: using roulette wheel selection [59,61,62], niching [63–65], and sigma method [66].

Undoubtedly, the most popular multi-objective algorithm is Non-dominated Sorting Genetic Algorithm version 2 (NSGA-II) [67,68]. This algorithm sorts the individuals based on their domination level. This means that non-dominated solutions are assigned rank 1, individuals that are dominated by the rank one solutions are assigned rank two, and so on. The selection and cross over are then done between the individuals ranked. The lower the rank number, the higher probability of selection and participation in generating the new population. NSGA-II was also equipped with a new niching operator to maintain high distribution of non-dominated solutions obtained across all objectives.

2.5. Contributions of the work

Despite the merits of the single-objective and multi-objective algorithms mentioned above, the NFL theorem says that none of them is able to solve *all* optimization problems [29]. This means that there is always possibility that a new-comer algorithm shows superior results on the current and/or new problems. In the literature no work simulates the behaviour of salp swarm in nature while salps form the biggest swarms on the planet and efficiently navigate and forage in the ocean. NFL and the lack of salp-inspired algorithm in the literature are the main motivations of the current work. It is worth mentioning here that the author has proposed a number of algorithms including Moth-Flame Optimization (MFO) [69] and GWO [15] recently. The algorithm proposed in this work is completely different in terms of inspiration, mathematical formulation, and real-world application. The MFO algorithm mimics the navigation of moths in nature and GWO simulates the hunting method of grey wolves, whereas the SSA algorithm is proposed based on the swarming behaviour of salps for the first time in the literature. In addition, the inspiration and mathematical model of this work completely differ from other publications written by the author in [70–73]. The following section discusses the main inspiration of the work, proposes mathematical models to simulate a salp swarm, and introduces two optimization algorithms for solving optimization problems with single of multiple objectives.

3. Inspiration, mathematical model, and Salp Swarm Algorithm

3.1. Inspiration

Salps belong to the family of Salpidae and have transparent barrel-shaped body. Their tissues are highly similar to jelly fishes. They also move very similar to jelly fish, in which the water is pumped through body as propulsion to move forward [74]. The shape of a salp is shown in Fig. 1(a).

The biological researches about this creature is in its early milestones mainly because their living environments are extremely difficult to access, and it is really difficult to keep them in laboratory environments. One of the most interesting behaviour of salps, which is of interest in the paper, is their swarming behaviour. In deep oceans, salps often form a swarm called salp chain. This chain is illustrated in Fig. 1(b). The main reason of this behaviour is not very clear yet, but some researchers believe that this is done for achieving better locomotion using rapid coordinated changes and foraging [75].

3.2. Proposed mathematical model for moving salp chains

There is little in the literature to mathematically model the swarming behaviours [76] and population of salps [77]. In addition, there is no mathematical model of salp swarms for solving optimization problems while swarms of bees, ants, and fishes have been widely modelled and used for solving optimization problems.

This subsection proposes the first model of salp chains in the literature for the purpose of solving optimization problems.

To mathematically model the salp chains, the population is first divided to two groups: leader and followers. The leader is the salp at the front of the chain, whereas the rest of salps are considered as followers. As the name of these salps implies, the leader guides swarm and the followers follow each other (and leader directly of indirectly).

Similarly to other swarm-based techniques, the position of salps is defined in an n-dimensional search space where n is the number of variables of a given problem. Therefore, the position of all salps are stored in a two-dimensional matrix called x. It is also assumed that there is a food source called F in the search space as the swarm's target.

To update the position of the leader the following equation is proposed:

$$x_j^1 = \begin{cases} F_j + c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right) & c_3 \geq 0 \\ F_j - c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right) & c_3 < 0 \end{cases} \tag{3.1}$$

where x_j^1 shows the position of the first salp (leader) in the jth dimension, F_j is the position of the food source in the jth dimension, ub_j indicates the upper bound of jth dimension, lb_j indicates the lower bound of jth dimension, c_1, c_2, and c_3 are random numbers.

Eq. (3.1) shows that the leader only updates its position with respect to the food source. The coefficient c_1 is the most important parameter in SSA because it balances exploration and exploitation defined as follows:

$$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \tag{3.2}$$

where l is the current iteration and L is the maximum number of iterations.

The parameter c_2 and c_3 are random numbers uniformly generated in the interval of [0,1]. In fact, they dictate if the next position in jth dimension should be towards positive infinity or negative infinity as well as the step size.

To update the position of the followers, the following equations is utilized (Newton's law of motion):

$$x_j^i = \frac{1}{2}at^2 + v_0 t \tag{3.3}$$

were $i \geq 2$, x_j^i shows the position of ith follower salp in jth dimension, t is time, v_0 is the initial speed, and $a = \frac{v_{final}}{v_0}$ where $v = \frac{x - x_0}{t}$.

Because the time in optimization is iteration, the discrepancy between iterations is equal to 1, and considering $v_0 = 0$, this equation can be expressed as follows:

$$x_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right) \tag{3.4}$$

where $i \geq 2$ and x_j^i shows the position of ith follower salp in jth dimension.

With Eqs. (3.1) and (3.4), the salp chains can be simulated.

3.2. Swarm simulation

In order to see the effects of the above mathematical model proposed, a simulation is done in this subsection. Twenty salps are randomly placed on a search space with stationary or moving sources of food. The position of the salp chains and history of each salp are drawn in Figs. 2–5. Note that the blue point in the figures shows the position of food source and the darkest filled circle is the leading salp. The follower salps are coloured with grey based on their position in the salp chain with respect to the leader. Inspecting the behaviour of salp chain over nine consecutive iterations in Figs. 2 and 4, it may be observed that the swarm can be formed and moved using the equation proposed effectively right after the first iteration. Also, it can be seen that the leading salp changes its position around the food source and follower salps gradually follow it over the course of iterations. The same model has been utilized for both simulations and the merits of the model proposed in both 2D and 3D spaces are evident in Figs. 2 and 4. It can be stated that the model is able to show the same behaviour in an n-dimensional space.

Figs. 3 and 5 show the position history of salps around a stationary and mobile food sources in 2D and 3D space after 100 iterations. The points searched around the stationary food source show that the salps effectively move around the search space. The distribution of points is reasonable and show that the model proposed is able to explore and exploit the space around the stationary food source. Also, Figs. 3 and 5 show that the mathematical model proposed requires salps in the salp chain to chase a moving food source. The distribution of the points searched around the start point is higher than the end point. This is due to the c_1 parameter which controls exploration and exploitations. These findings evidence that the model of salp chain movement is able to explore and exploit the space around both stationary and mobile food sources.

3.3. Single-objective Salp Swarm Algorithm (SSA)

Needless to say, the mathematical model for simulating salp chains cannot be directly employed to solve optimization problems. In other words, there is a need to tweak the model a little bit to make it applicable to optimization problems. The ultimate goal of a single-objective optimizer is to determine the global optimum. In the SSA swarm model, follower salps follow the leading salp. The leading salp also moves towards the food source. If the food source be replaced by the global optimum, therefore, the salp chain automatically moves towards it. However, the problem is that the global optimum of optimization problems is unknown. In this case, it is assumed that the best solution obtained so-far is the global optimum and assumed as the food source to be chased by the salp chain.

The pseudo code of the SSA algorithm is illustrated in Fig. 6.[1] This figure shows that the SSA algorithm starts approximating the global optimum by initiating multiple salps with random positions. It then calculates the fitness of each salp, finds the salp with the best fitness, and assigns the position of the best salp to the variable F as the source food to be chased by the salp chain. Meantime the coefficient c_1 is updated using Eq. (3.2). For each dimension, the position of leading salp is updated using Eq. (3.1) and the position of follower salps are updated utilizing Eq. (3.4). If any of the salp goes outside the search space, it will be brought back on the boundaries. All the above steps except initialization are iteratively executed until the satisfaction of an end criterion.

It should be noted that the food source will be updated during optimization because the salp chain is very likely to find a better solution by exploring and exploiting the space around it. The simulations in the Section 3.2 show that the salp chain modelled is able to chase a moving food source. Therefore, the salp chain has the potential to move towards the global optimum that changes over the course of iterations. To see how the proposed salp chain model and SSA algorithm are effective in solving optimization problems, some remarks are listed as follows:

[1] The source code of the SSA algorithm is publicly available at https://au.mathworks.com/matlabcentral/fileexchange/63745-ssa--salp-swarm-algorithm and http://www.alimirjalili.com/Projects.html.

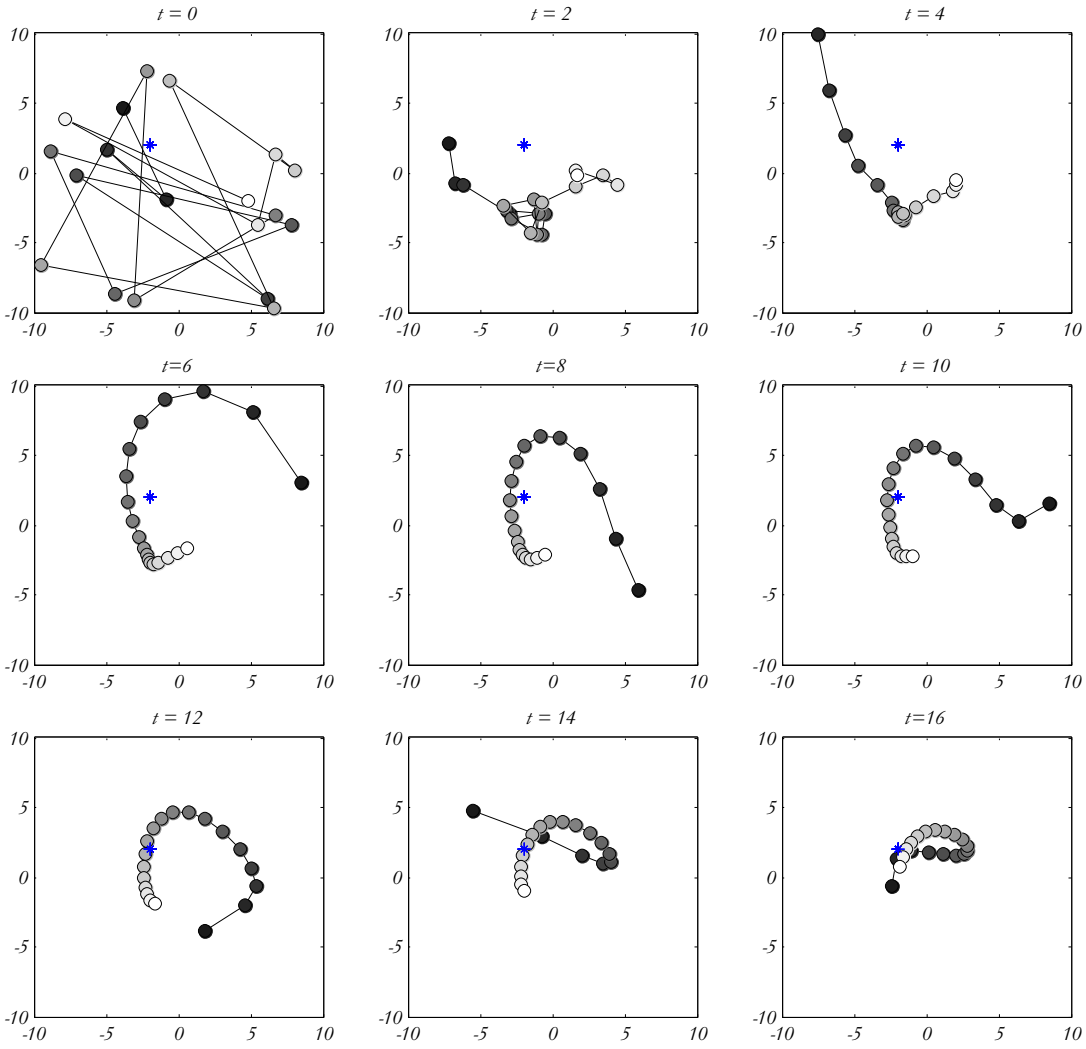

Fig. 2. Slap chain movement around a stationary source of food in a 2D space.

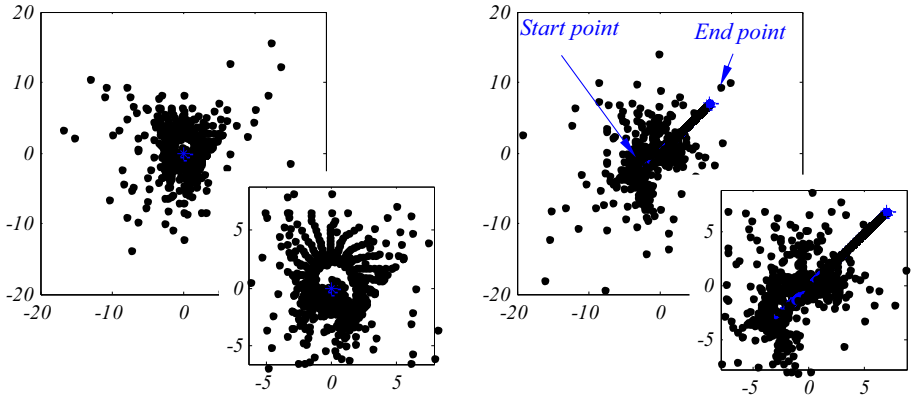

Fig. 3. Search history around stationary and mobile food sources in a 2D space after 100 iterations.

- SSA algorithm saves the best solution obtained so far and assigns it to the food source variable, so it never get lost even if the whole population deteriorates.
- SSA algorithm updates the position of the leading salp with respect to the food source only, which is the best solution obtained so far, so the leader always explores and exploits the space around it.
- SSA algorithm updates the position of follower salps with respect to each other, so they move gradually towards the leading salp.
- Gradual movements of follower slaps prevent the SSA algorithm from easily stagnating in local optima.
- Parameter c_1 is decreased adaptively over the course of iterations, so the SSA algorithm first explores the search space and then exploits it.

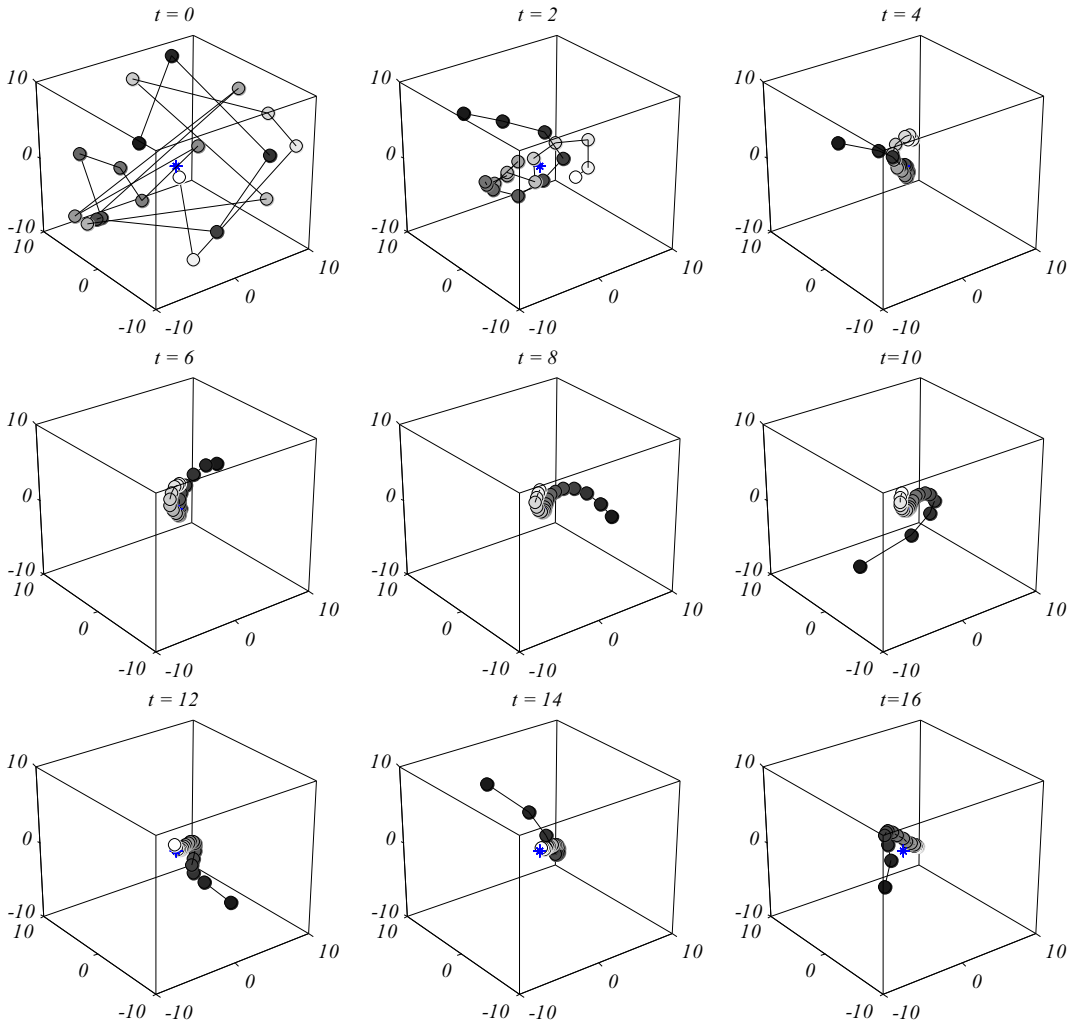

Fig. 4. Slap chain movement around a stationary source of food in a 3D space.

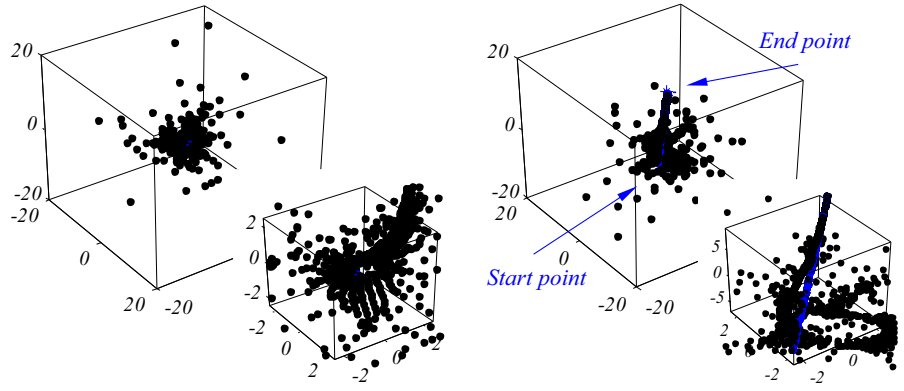

Fig. 5. Search history around stationary and mobile food sources in a 3D space after 100 iterations.

- SSA algorithm has only one main controlling parameter (c_1).
- SSA algorithm is simple and easy to implement.

These remarks make the SSA algorithm theoretically and potentially able to solve single-objective optimization problems with unknown search spaces. The adaptive mechanism of SSA allows this algorithm to avoid local solutions and eventually finds an accurate estimation of the best solution obtained during optimization. Therefore, it can be applied to both unimodal and multi-modal problems. The above-mentioned advantages potentially allow SSA to outperform recent algorithms (MFO, GWO, ABC, etc). However, this cannot be guaranteed for all optimization problems according to the NFL theorem.

Note that the computational complexity of the SSA algorithm is of $O(t(d^*n + Cof^*n))$ where t shows the number of iterations, d is the number of variables (dimension), n is the number of solutions, and Cof indicates the cost of objective function. In Section 4, these claims are investigated experimentally on both benchmark and real-world problems.

```
Initialize the salp population x_i (i = 1, 2, ..., n) considering ub and lb
while (end condition is not satisfied)
Calculate the fitness of each search agent (salp)
F=the best search agent
Update c_1 by Eq. (3.2)
    for each salp (x_i)
        if (i==1)
            Update the position of the leading salp by Eq. (3.1)
        else
            Update the position of the follower salp by Eq. (3.4)
        end
    end
    Amend the salps based on the upper and lower bounds of variables
end
return F
```

Fig. 6. Pseudo code of the SSA algorithm.

3.4. Multi-objective Salp Swarm Algorithm (MSSA)

As discussed in Section 2, the solution for a multi-objective problem is a set of solutions called the Pareto optimal set. The SSA algorithm is able to drive salps towards the food source and updates it over the course of iterations. However, this algorithm is not able to solve multi-objective problems mainly due to the following two reasons:

- SSA only saves one solution as the best solution, so it cannot store multiple solutions as the best solutions for a multi-objective problem.
- SSA updates the food source with the best solution obtained so far in each iteration, but there is no single best solution for multi-objective problems.

This first issue is tackled by equipping the SSA algorithm with a repository of food sources. This repository maintains the best non-dominated solutions obtained so far during optimization and is very similar to the archives in Multi-Objective Particle Swarm Optimization (MOPSO) [78]. The repository has a maximum size to store a limited number of non-dominated solutions. During optimization, each salp is compared against all the repository residents using the Pareto dominance operators. If a salp dominates a solution in the repository, they have to be swapped. If a salp dominates a set of solutions in the repository, they all should be removed from the repository and the salp should added in the repository. If at least one of the repository residents dominates a salp in the new population, it should be discarded straight away. If a salp is non-dominated in comparison with all repository residents, it has to be added to the archive.

These rules can guarantee that the repository always stores the non-dominated solutions obtained so far by the algorithm. However, there is a special case where the repository becomes full and a salp is non-dominated in comparison with the repository residents. Of course, the easiest way is to randomly delete one of the solutions in the archive and replace it with the non-dominated salp. A wiser way is to remove one of the similar non-dominated solutions in the repository. Since *a posteriori* multi-objective algorithm should be able to find uniformly distributed Pareto optimal solutions, the best candidate to remove from the archive is the one in a populated region. This approach improves the distribution of the archive residents over the course of iterations.

To find the non-dominated solutions with populated neighbourhood, the number of neighbouring solutions with a certain maximum distance is counted and assumed. This distance is defined by $\vec{d} = \frac{\vec{max} - \vec{min}}{repository\ size}$ where *max* and *min* are two vectors for storing maximum and minimum values for every objective respectively. The repository with one solution in each segment is the best case. After assigning a rank to each repository resident based on the number of neighbouring solutions, a roulette wheel is employed to choose one of them. The more number of neighbouring solutions

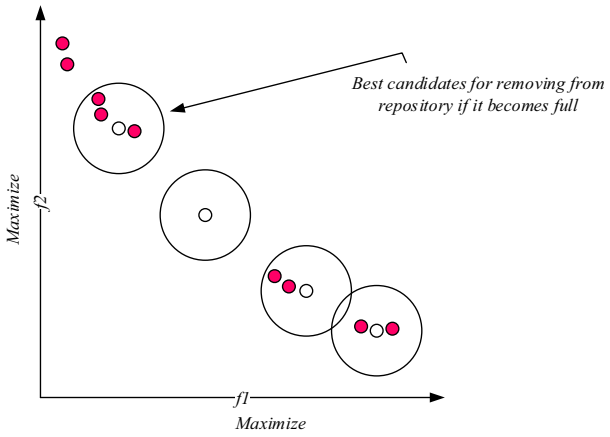

Fig. 7. Update mechanism of the repository when it is full.

```
Initialize the salp population x_i (i = 1, 2, ..., n) considering ub and lb
while (end criterion is not met)
    Calculate the fitness of each search agent (salp)
    Determine the non-dominated salps
    Update the repository considering the obtained non-dominated salps
    if the repository becomes full
        Call the repository maintenance procedure to remove one repository resident
        Add the non-dominated salp to the repository
    end
    Choose a source of food from repository: F=SelectFood(repository)
    Update c_1 by Eq. (3.2)
    for each salp (x_i)
        if (i==1)
            Update the position of the leading salp by Eq. (3.1)
        else
            Update the position of the follower salp by Eq. (3.4)
        end
    end
    Amend the salps based on the upper and lower bounds of variables
end
return repository
```

Fig. 8. Pseudo code of the MSSA algorithm.

(the larger rank number) for a solution, the higher probability of removing it from the repository. An example of this repository update mechanism is illustrated in Fig. 7. Note that the neighbourhood should be defined for all the solutions, but only four of the non-dominated solutions are investigated in this figure.

As mentioned above, the second issue when solving multi-objective problems using SSA is the selection of the food source because there is more than one best solution in a multi-objective search space. Again, the food source can be chosen randomly from the repository. However, a more appropriate way is to select it from a set of non-dominated solutions with the least crowded neighbourhood. This can be done using the same ranking process and roulette wheel selection employed in the repository maintenance operator. The main difference is the probability of choosing the non-dominated solutions. In the archive maintenance, the solutions with higher rank (crowded neighbourhood) are more likely to be chosen. By contrast, the less populated neighbourhood (the lower rank number) for a non-dominated solution in the repository, the higher probability of being selected as the food source. In Fig. 7 for instance, the non-dominated solutions in the middle with no neighbouring solution has the highest probability to be chosen as the food source. After all, the pseudo code of the Multi-objective Salp Swarm Algorithm (MSSA) is shown in Fig. 8.[2]

[2] The source code of the MSSA algorithm is publicly available at https://au.mathworks.com/matlabcentral/fileexchange/63746-mssa--multi-objective-salp-swarm-algorithm and http://www.alimirjalili.com/Projects.html.

Fig. 8 shows that the MSSA algorithm first initializes the population of salps with respect to upper bounds and lower bourns of variables. This algorithm then calculates the objective values for each salps and finds the non-dominated ones. The non-dominated solutions are added to the archive if the repository is not full. If the repository is full, the repository maintenance is run to delete the solutions with crowded neighbourhood. In this step, the solutions are first ranked and then selected using a roulette wheel. After removing enough number of repository residents, the non-dominated salps can be added to the repository. After updating the repository, a food source is selected from the non-dominated solutions in the repository with the least crowded neighbourhood. Similarly to the archive maintenance, this is done by ranking the solutions and employing a roulette wheel. The next step is to update c_1 using Eq. (3.2) and update the position of leading/follower salps using either Eq. (3.1) or (3.4). If during updating position process a salp goes outside of the boundaries, it will be brought back on the boundary. Finally, all the above steps except initialization are repeated until the satisfaction of an end condition.

To see how effective the MSSA algorithm is, some comments are:

- The non-dominated solutions obtained so far are stored in a repository, so they never get lost even if the entire population deteriorates in an iteration.
- The solutions with crowded neighbourhood are discarded every time the repository maintenance is called, which results in improving the coverage of non-dominated solutions across all objectives.
- A food source is selected from the list of non-dominated solutions with the least number of neighbouring solutions, which leads the search towards the less crowded regions of the Pareto optimal front obtained and improves the coverage of solutions found.
- MSSA inherits the operators of SSA due to the use of a similar population division (leading and follower salps) and position updating process.
- MSSA algorithm has only two main controlling parameter (c_1 and archive size).
- MSSA algorithm is simple and easy to implement.

These comments show that the MSSA algorithm is logically able to find accurate Pareto optimal solutions with high distribution across all objectives. Note that the computational complexity of MSSA algorithm is of $O(t(d*n + Cof*n + M*n^2))$ where M indicates the number of objectives t shows the number of iterations, d is the number of variables (dimension), n is the number of solutions, and Cof indicates the cost of objective function. In the next sections, these statements are investigated and proved experimentally on benchmark and real problems.

4. Results

For proving the theoretical claims pointed out in the previous sections, a variety of experiments are conducted. Firstly, a set of qualitative metrics is employed to find out if SSA is better than other similar algorithms. Secondly, the quantitative results are collected to measure how much better the SSA is compared to similar algorithms. Finally, the MSSA algorithm is compared to similar algorithms in the literature. Note that several challenging single- and multi-objective benchmark problems are utilized in this section. The following subsections present the experimental set up for each of the experimental phases, present the results, and discuss them in detail.

4.1. Qualitative results of SSA and discussion

Qualitative results are mostly derived from the different visualization tools. The most common qualitative results in the literature of single-objective optimization are convergence curves. Researchers usually store the best solution obtained so far in each iteration and draw them as a line to be able to observe how well an algorithm improves the approximation of the global optimum over the course of iterations. In this study, several other qualitative results are utilized as well to clearly observe the performance of SSA when solving different problems.

To generate the qualitative results, the first step is to find a suitable test bed. We definitely need a set of test cases to test SSA and observe its performance qualitatively. It is always beneficial to employ a set of standard test beds with different characteristics to challenge and benchmark different abilities of an algorithm. Standard test beds are designed to challenge algorithms in a standard way and obviously allow us to conveniently and confidently compare an algorithm with others. Diversity of test functions allows observing and testing the ability of algorithms from different perspectives.

The current benchmark functions can be grouped to three main families: unimodal, multi-modal, and composite [7,79–81]. The landscapes of these classes of test problems are illustrated in Fig. 9. It may be observed in this figure that unimodal test functions have only one optimum and there are no local optima. These types of search spaces are appropriate for testing the convergence speed and exploitive behaviour. Multi-modal and composite benchmark functions have more than one optimum, which make them suitable for benchmarking the local optima avoidance and explorative behaviour of optimization algorithms. Composite test functions are usually more challenging than multi-modal ones and highly similar to real search spaces. In this subsection, eight of these test functions are chosen as case studies. The mathematical formulations and shapes of the test functions are presented in the Appendix A and Fig. 10 respectively.

The qualitative results are collected by different qualitative metrics. The first qualitative results are search histories of search agents in SSA over the course of iteration. Search history figures usually show the position history of all agents during optimization. With saving and illustrating the positions history, we can observe the sampled regions of the search space by an algorithm and the probable search patterns in the entire swarm. In a search history figure, we can see if and how an algorithm explores and exploits the search space as well. However, we cannot see the order in which the algorithm performs exploration and exploitation. Therefore, we need another qualitative metric. The search history of the SSA algorithm when solving all the test functions are illustrated in Fig. 10.

Inspecting the search histories in Fig. 10, it is evident that the SSA algorithm samples the most promising regions of the search space. This pattern can be observed in unimodal, multimodal, and composite test functions. In unimodal test functions the distribution of sampled point is sparse in the non-promising regions, while more sampled points can be seen around the search space in multi-modal and composite benchmark functions. This is because of the difficulty of the multi-modal and composite benchmark functions and shows how well SSA is able to bias the search towards promising regions of the search space proportional to the level of difficulty of the problem.

Also, this behaviour can support the local optima avoidance and exploration of the SSA algorithm. On the other hand, the distribution of sampled points is high around the true global optimum on all of the test functions. This observation is able to support exploitation and convergence of the SSA algorithm towards to true global optimum. However, there is a need for more experiments

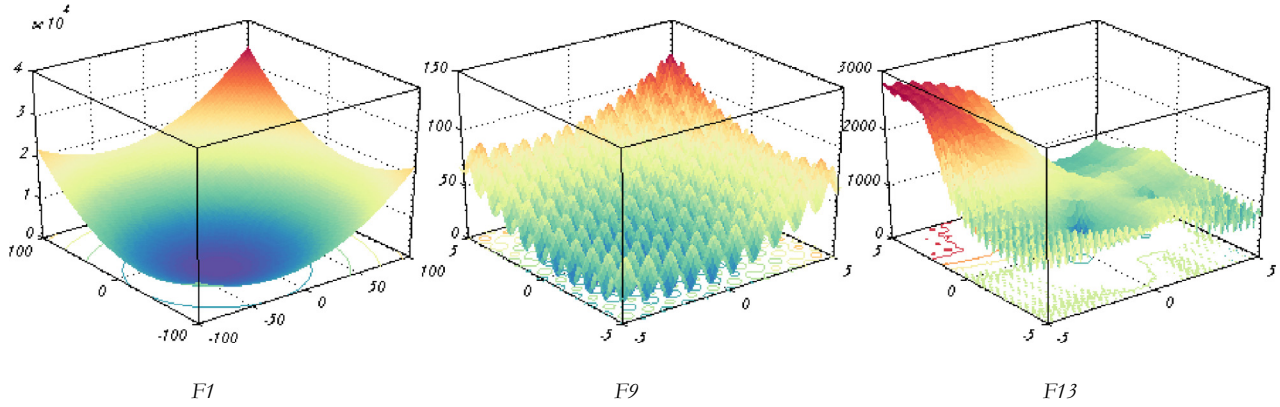

Fig. 9. An example of unimodal test function, multi-modal test function, and composite test function.

to observe if the exploration occurs before exploitation and if this assists SSA to improve the accuracy of the approximated global optimum. Such analyses are covered in the following paragraphs.

The second qualitative results are collected from the trajectory of search agents. Since showing the trajectory of all search agents across all dimensions results in occlusion and difficulty of analysis, the trajectory of the first salp in the first dimension is only stored and illustrated. These results are collected to observe and confirm in which order SSA performs exploration and exploitation. The trajectory curves when solving the benchmark functions are depicted in Fig. 10. This figure shows that the first salp faces sudden changes in the initial steps of iterations, gradual changes after the initial steps, and monotonous behaviour in the final steps. This proves that salps move abruptly at the beginning and tend to fluctuate gradually proportional to the number of iterations. As per this observation, SSA first requires the salps to go around the search space and causes exploration of the search space. Consecutively, SSA exploits the search space by driving the salps towards the global optimum and encouraging them to move locally instead of globally.

The search history and trajectory figures show how SSA explores and exploits the search space. However, such results do not show if exploration and exploitation are beneficial in terms of improving the first random population and finding an accurate approximation (improving) of the global optimum, which are the main objectives for all optimization algorithms. In order to observe and confirm such behaviours, average fitness of all salps and the best solution found so far (convergence curve) are saved during the optimization process and illustrated in Fig. 10. The curves for the average fitness of salps show that there is a descending trend on all the test functions. Since all of the test functions are minimization problems, these results prove that the SSA is able to improve the quality of the population proportional to the number of iterations. Another interesting pattern is the deterioration of the fitness of the population especially in the initial steps of optimization. This is due to the effect of exploration, in which salps just churn around the space. However, the average fitness curve smoothly decreases proportional to the iteration number, which is because of the exploitation phase.

Of course the average fitness curves show how well SSA improve the population during optimization, it cannot be stated that the approximation of the global optimum also increases. This is where the convergence diagrams are beneficial. Inspecting the convergence curves in Fig. 10, it is evident that the fitness of the approximation of the global optimum obtained in each iteration is improved by the SSA algorithm over the course of iterations. This improvement is not consistent always as can be seen in some of the figures. This demonstrates that SSA shows different convergence behaviours when solving different problems. For instance, the convergence curve is very smooth and steady on the umimodal test functions, which shows that SSA benefits from high exploitation and convergence. In multi-modal and composite test functions, however, SSA shows no improvements for some iterations. This is because SSA emphasises exploration in such problems, which sometimes results in sampling non-promising regions but boosting local optima avoidance.

The above qualitative results showed that SSA first explores the search space and then exploits it. In addition, the results proved that SSA can sample different regions of the search space very effectively by coving promising regions of the search space. It was observed that SSA is capable of improving the quality of a set of random solutions for a given problem. Finally, the results and discussion showed that the accuracy of the approximated global optimum is increased by SSA, which is the outcome of a proper balance of exploration/local optima avoidance and exploitation/ convergence.

4.2. Quantitative results of SSA and discussion

Although the qualitative results proved high exploration and exploitation of the SSA algorithm, they cannot show how much good this algorithm is. This section employs two performance indicators to quantify the performance of the SSA algorithm and compares it with other similar algorithms in the literature. The performance metrics employed are average and standard deviation of the best solutions obtained found in 30 independent runs. The former performance metric shows how the SSA algorithm performs in average, whereas the latter one indicates how stable SSA is during all the runs. Although these two indicators are able to measure the overall performance of SSA, they cannot measure and compare each of the runs individually.

In order to compare each of the runs and make sure about the significance of the results, therefore, the Wilcoxon rank-sum test is conducted in this subsection as well. The *p*-values that are less than 0.05 could be considered as strong evidence against the null hypothesis. For the statistical test, the best algorithm in each test function is chosen and compared with other algorithms independently. For example, if the best algorithm is SSA, the pairwise comparison is done between SSA/PSO, SSA/GSA, and so on. The same approach is followed throughout the paper.

The same test functions utilized in the previous section as well as a few more are employed here (19 in total). However, the number of dimensions is increased to 20. This has been done to benchmark the performance of SSA in solving challenging problems with a large number of variables. Note that the details of these test functions are available in the Appendix A. To verify the results, the

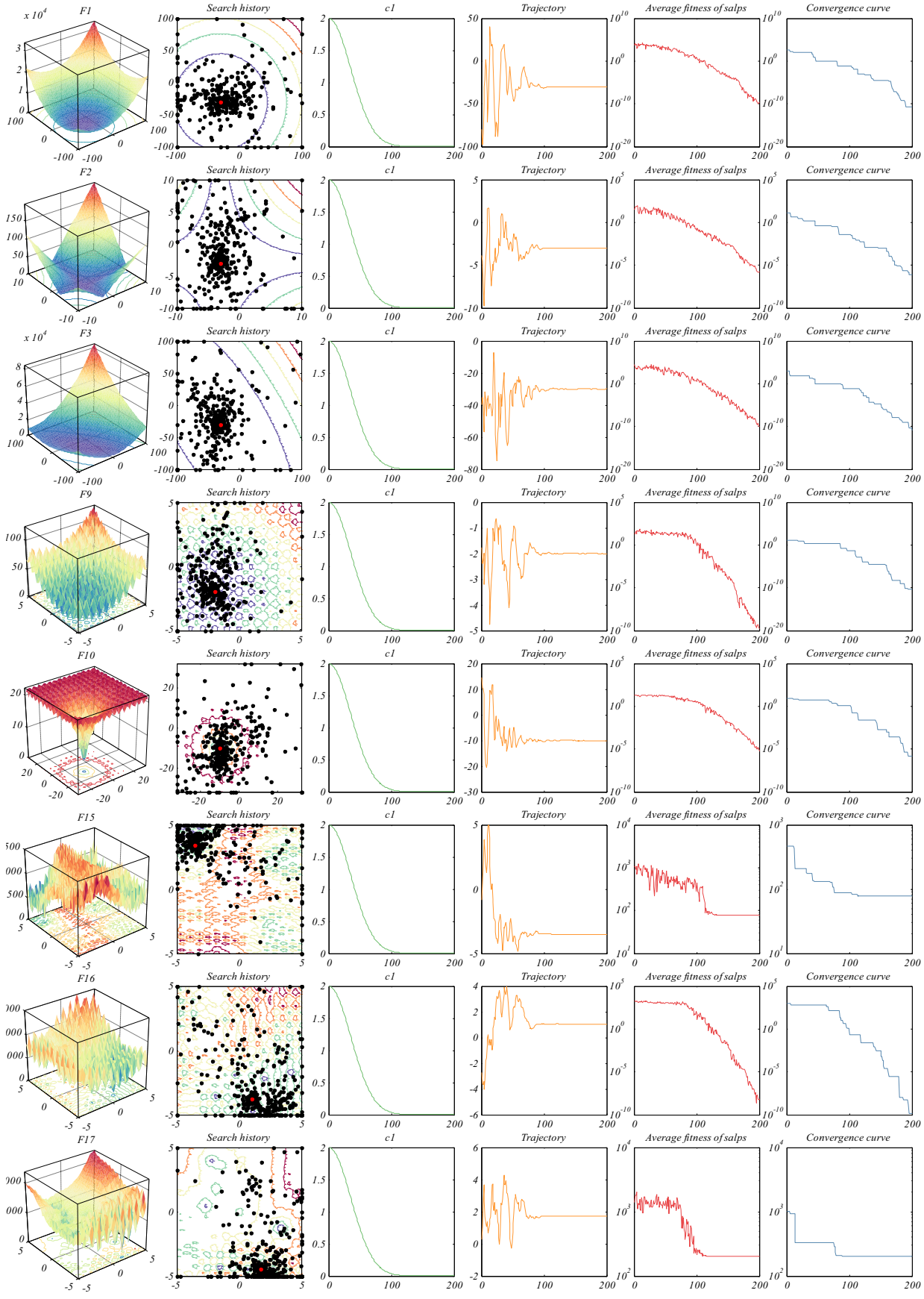

Fig. 10. Qualitative results: search history, trajectory, average fitness, and convergence.

Table 1
Results of algorithms on the unimodal test functions.

F	SSA		PSO		GSA		BA	
	Ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F1	0.0000	0.0000	0.2148	0.2663	0.0030	0.0224	0.7518	1.0000
F2	0.2272	1.0000	0.2858	0.0867	0.0000	0.0084	1.0000	0.4826
F3	0.0000	0.0000	0.1502	0.1436	0.0289	0.0374	1.0000	1.0000
F4	0.0000	0.6556	0.3443	0.1023	0.1821	0.0981	0.9059	1.0000
F5	0.0000	0.0000	0.0461	0.0706	0.0000	0.0000	1.0000	1.0000
F6	0.0000	0.0000	0.7212	0.5303	0.3944	0.2328	1.0000	1.0000
F7	0.0028	0.0070	0.0817	0.0635	0.0017	0.0058	1.0000	1.0000
F	FPA		SMS		FA		GA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F1	0.0240	0.0388	1.0000	0.3990	0.1407	0.0845	0.3485	0.3714
F2	0.5394	0.3095	0.5152	0.1338	0.3064	0.0000	0.4172	0.0764
F3	0.0034	0.0008	0.2049	0.0508	0.0419	0.0079	0.1049	0.0362
F4	0.1571	0.2651	1.0000	0.2571	0.2882	0.0000	0.6959	0.0174
F5	0.0031	0.0054	0.7049	0.4865	0.0095	0.0053	0.1056	0.0836
F6	0.0153	0.0561	0.7540	0.3097	0.0410	0.0223	0.1892	0.3023
F7	0.0138	0.0140	0.0000	0.0000	0.0445	0.0274	0.3625	0.1503

Table 2
Results of multimodal benchmark functions.

F	SSA		PSO		GSA		BA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F8	1.0000	0.0071	1.0000	0.0094	1.0000	0.0026	0.0000	1.0000
F9	0.4254	0.9502	0.3548	0.6283	0.0000	0.3290	0.6155	1.0000
F10	0.0598	0.5279	0.5917	0.9783	0.0000	0.0000	0.9443	0.4541
F11	0.0000	0.0000	0.8481	0.6827	1.0000	0.4911	0.9757	1.0000
F12	0.0000	0.0000	0.0714	0.0572	0.0000	0.0000	1.0000	1.0000
F13	0.0000	0.0000	0.0962	0.0911	0.0000	0.0001	1.0000	1.0000
F	FPA		SMS		FA		GA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F8	1.0000	0.0021	1.0000	0.0000	1.0000	0.0009	1.0000	0.0016
F9	0.5894	0.6006	0.9074	0.5564	0.8299	0.1451	1.0000	0.0000
F10	0.7708	1.0000	1.0000	0.2696	0.6937	0.4449	0.8864	0.2961
F11	0.0334	0.1125	0.6303	0.8695	0.0492	0.0327	0.1728	0.1934
F12	0.0000	0.0000	0.2330	0.0275	0.0001	0.0001	0.0297	0.0252
F13	0.0011	0.0074	0.0939	0.0000	0.0008	0.0009	0.0335	0.0438

SSA algorithm is compared against a collection of well-known and recent algorithms: PSO, GSA, BA, FPA, SMS, FA, and GA. To provide a fair comparison, the main controlling parameters of these algorithms, number of search agents and maximum iteration, are equal to 30 and 500 respectively. For other controlling parameters of each algorithm, the values in the latest version (source code) are used to ensure the best performance. Each of the algorithms are run 30 times on each of the test functions and the results are presented in Tables 1–4. Note that the results of the algorithms are normalized in [0,1] using the min-max normalization to be able to compare their performances on different test functions.

The results on the unimodal functions show that the SSA algorithm outperforms other algorithms in the majority of the test functions. The better mean values show that SSA performs better than others in average, and the standard deviations prove that this superiority is stable. The results of the *p*-values in Table 4 generated from the Wilcoxon test show that the superiority is statistically significant. Due to the presence of single optimum in the unimoal in unimodal test functions, they can benchmark only exploitation and convergence of algorithms. Therefore, these results demonstrate that SSA benefits from high exploitation and convergence speed.

Inspection of the results on the multi-modal test functions in Table 2, it may be observed that the SSA algorithm again outperforms other algorithms on most of the test functions. The results of Wilcoxon statistical test prove that the results are statistically significant because most of the *p*-values are less than 0.05. The better results can be seen in both average and standard deviation, which indicate how well and robust SSA is when solving such problems. In contrast to the unimodal test functions, multi-modal ones have many optima, in which one of them is the global and the rest are local. The results of the SSA algorithm on these case studies prove that this algorithm can explore the search space efficiently. This high exploration of SSA causes avoiding the many number of local optima in a multi-modal search space.

As can be seen in the results of the algorithms on composite benchmark functions in Table 3, SSA is able to show very competitive results in these case studies as well. The average and standard deviation of the best solution obtained during 30 runs testify that this algorithm shows superior and steady performance in average. The *p*-values also support the better results of SSA on the composite test functions and prove how significant this algorithm is. Composite functions are the combination of many different test functions and offer very challenging test beds. This makes them very similar to the real search space that SSA algorithm might face when solving a practical problem. Solving such problems requires a very well-timed and well-tuned exploration and exploita-

Table 3
Results of algorithms on the composite test functions.

F	SSA		PSO		GSA		BA	
	Ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F14	0.0557	0.8090	0.4179	1.0000	0.1160	0.5572	1.0000	0.6024
F15	0.0000	0.0000	0.4081	0.8317	0.1224	0.5715	1.0000	1.0000
F16	0.1952	0.1527	0.6181	0.5347	0.0000	0.0000	1.0000	1.0000
F17	0.0000	0.0651	0.4694	0.8406	0.1752	0.9897	1.0000	0.9450
F18	0.1417	0.5571	0.3566	0.7841	0.0747	0.1599	1.0000	1.0000
F19	0.0832	0.7059	0.6883	1.0000	0.9811	0.0000	1.0000	0.2160
F	FPA		SMS		FA		GA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F14	0.0000	0.0957	0.5304	0.4176	0.4549	0.0000	0.5656	0.1349
F15	0.0176	0.0387	0.4554	0.5404	0.5546	0.4342	0.1868	0.0448
F16	0.3158	0.1006	0.7308	0.2338	0.4585	0.1002	0.5721	0.2255
F17	0.0859	0.0000	0.6337	0.3146	0.4893	1.0000	0.3465	0.0398
F18	0.0000	0.0000	0.2885	0.4348	0.2397	0.3315	0.1360	0.1966
F19	0.0000	0.0860	0.2839	0.8509	0.8001	0.8833	0.0773	0.0430

Table 4
p-Values obtained from the rank-sum test for the results in Table 1–3 (N/A stands for not applicable).

F	SSA	PSO	GSA	BA	FPA	SMS	FA	GA
F1	N/A	0.000183	0.472676	0.000183	0.000183	0.000183	0.000183	0.000183
F2	0.007285	0.000183	N/A	0.000183	0.000183	0.000183	0.000183	0.000183
F3	N/A	0.000183	0.000183	0.000183	0.000583	0.000183	0.000183	0.000183
F4	N/A	0.000183	0.014019	0.000183	0.031209	0.000183	0.00044	0.000183
F5	0.850107	0.000183	N/A	0.000183	0.000183	0.000183	0.000183	0.000183
F6	N/A	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F7	0.472676	0.000183	0.677585	0.000183	0.000583	N/A	0.000183	0.000183
F8	0.000183	0.000183	0.000183	N/A	0.000183	0.000183	0.000183	0.000183
F9	0.001706	0.005795	N/A	0.000246	0.000183	0.000183	0.000183	0.000183
F10	0.000183	0.000183	N/A	0.000183	0.000183	0.000183	0.000183	0.000183
F11	N/A	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F12	N/A	0.000183	0.000183	0.000183	0.00044	8.74E-05	0.000183	0.000183
F13	N/A	0.000183	0.000183	0.000183	0.000183	6.39E-05	0.000183	0.000183
F14	0.10411	0.045155	0.790566	0.000183	N/A	0.00044	0.00044	0.000183
F15	N/A	0.002202	0.472509	0.000183	0.212294	0.000183	0.000183	0.000183
F16	0.000583	0.000183	N/A	0.000183	0.000183	0.000183	0.000183	0.000183
F17	N/A	0.000769	0.57075	0.000183	0.037635	0.000183	0.00033	0.000183
F18	0.140465	0.002827	0.025748	0.000183	N/A	0.000183	0.000183	0.000183
F19	0.021134	0.005795	0.000183	0.000183	N/A	0.000183	0.000183	0.000183

tion. Therefore, these results and findings demonstrate that the SSA algorithm is able to solve very challenging problems as well.

It is worth discussing here that the exploitation of SSA tends to be better than its exploration. This can be inferred from the results of this algorithm on unimodal test functions compared to multi-modal ones. This is due to the swarm-based nature of this algorithm, in which abrupt changes in the solutions are lower than evolutionary algorithms with crossover operators. Despite this fact, the results show that this is not a concern since the explorative behaviour of SSA is also good due to the position updating mechanism employed. In fact, mere exploration does not guarantee finding the global optimum, and a proper balance of exploration and exploitation is required.

4.3. Comparison of SSA with Harmony Search (HS) on CEC-BBOB-2015 test functions

In this subsection, the proposed SSA algorithm is compared to HS [21,82,83], PSO, GSA, and FPA. HS has been widely used [84–86] and applied to a significant number of problems in the literature. To better compare and challenge the SSA algorithm, a very recent test suite called CEC-BBOB-2015 [87] including noiseless [88] and noisy functions [89] is employed. Each algorithm is run 30 times with 30 solutions and 500 number of iterations. The results are shown in Table 5 and 6. Note that the results are normalized again to conveniently compare algorithm on different test functions.

Table 5 show that the proposed SSA algorithm is able to significantly outperform other algorithms on the majority of noiseless and noisy test functions in the CEC-BBOB-2015 test suite. The *p*-values in Table 6 suggest that the superiority of SSA is statistically significant. These results show that the proposed algorithm is able to solve highly challenging test functions with and without noise as well.

4.4. Scalability analysis

Since real-world problems often have a large number of variables, this subsection analyses the scalability of the proposed SSA algorithm. One unimodal (F1) and one multi-modal test functions (F10) with a varied number of parameters are employed. The number of parameters are changed from 50 to 200 with the step size of 10. SSA is run 30 times on each test function with different number of variables and the average of best solution obtained are presented in Fig. 11. This experiment is done with 200 solutions and 500 iterations.

Fig. 11 shows that the performance of SSA degrades when increasing the number of variables. This is expected since the number of solutions was fixed during this experiment. The main observation in this figure is that the performance of SSA does not significantly degrade when solving problems with a large number of

Table 5
Results of SSA, HS, PSO, GSA, and FPA on CEC-BBOB-2015 test suite.

Test function	SSA		HS		PSO		GSA		FPA	
	ave	std	ave	std	ave	std	ave	std	ave	std
CEC-BBOB-2015-F1	0.0000	0.0000	0.4122	0.2752	0.4357	0.3727	1.0000	1.0000	0.2459	0.2740
CEC-BBOB-2015-F2	0.0147	0.0099	0.0954	0.0431	0.1922	0.1603	1.0000	1.0000	0.0000	0.0000
CEC-BBOB-2015-F3	0.0000	0.0729	0.1776	0.0000	0.3985	0.3658	1.0000	1.0000	0.2490	0.1598
CEC-BBOB-2015-F4	0.0000	0.3758	0.2131	0.1996	0.3212	0.4457	1.0000	1.0000	0.4058	0.0000
CEC-BBOB-2015-F5	0.0000	0.3471	0.2791	0.0000	0.4836	0.8474	1.0000	1.0000	0.2821	0.3779
CEC-BBOB-2015-F6	0.0000	0.0000	0.0492	0.1036	0.2471	0.6694	1.0000	1.0000	0.0013	0.0006
CEC-BBOB-2015-F7	0.0000	0.0000	0.3713	0.1727	0.2234	0.2677	1.0000	1.0000	0.0640	0.0838
CEC-BBOB-2015-F8	0.0000	0.0000	0.5168	0.8007	0.4621	1.0000	1.0000	0.6090	0.1163	0.1232
CEC-BBOB-2015-F9	0.0000	0.0000	1.0000	1.0000	0.2347	0.4027	0.4038	0.9961	0.0391	0.0282
CEC-BBOB-2015-F10	0.0136	0.0205	1.0000	0.8395	0.2109	0.3116	0.5740	1.0000	0.0000	0.0000
CEC-BBOB-2015-F101	0.0000	0.0000	0.1917	0.1337	0.2362	0.2997	1.0000	1.0000	0.1623	0.2302
CEC-BBOB-2015-F102	0.0000	0.0000	0.2176	0.2725	0.3405	0.5386	1.0000	1.0000	0.2422	0.5889
CEC-BBOB-2015-F103	0.0000	0.0000	0.2298	0.1785	0.2440	0.5690	1.0000	1.0000	0.1887	0.4013
CEC-BBOB-2015-F104	0.0000	0.0000	0.1029	0.0722	0.1067	0.1153	1.0000	1.0000	0.0394	0.0489
CEC-BBOB-2015-F105	0.0000	0.0000	0.0988	0.1634	0.1052	0.3903	1.0000	1.0000	0.0548	0.1160
CEC-BBOB-2015-F106	0.0000	0.0000	0.3434	0.2254	0.3032	0.3099	1.0000	1.0000	0.1295	0.1709
CEC-BBOB-2015-F107	0.0353	0.1371	1.0000	1.0000	0.0000	0.0000	0.2668	0.3331	0.3639	0.1765
CEC-BBOB-2015-F108	0.0192	0.0360	1.0000	1.0000	0.0000	0.0000	0.0132	0.0416	0.0068	0.0281
CEC-BBOB-2015-F109	0.2047	0.9840	1.0000	0.9757	0.0865	1.0000	0.1860	0.7203	0.0000	0.0000
CEC-BBOB-2015-F110	0.0000	0.0000	1.0000	0.9979	0.1048	0.2387	0.7144	1.0000	0.2670	0.3995

Table 6
p-Values obtained from the rank-sum test for the results in Table 5 (N/A stands for not applicable).

Test function	SSA	HS	PSO	BA	FPA
CEC-BBOB-2015-F1	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F2	0.000769	0.000183	0.000183	0.000183	1.000000
CEC-BBOB-2015-F3	N/A	0.000440	0.000769	0.000183	0.000769
CEC-BBOB-2015-F4	N/A	0.002202	0.000440	0.000183	0.000183
CEC-BBOB-2015-F5	N/A	0.000183	0.000183	0.000183	0.000246
CEC-BBOB-2015-F6	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F7	N/A	0.000183	0.000183	0.000183	0.002827
CEC-BBOB-2015-F8	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F9	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F10	0.025748	0.000183	0.000183	0.000183	1.000000
CEC-BBOB-2015-F101	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F102	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F103	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F104	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F105	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F106	N/A	0.000183	0.000183	0.000183	0.000183
CEC-BBOB-2015-F107	0.909722	0.000183	N/A	0.004586	0.000183
CEC-BBOB-2015-F108	0.427355	0.000183	N/A	0.677585	0.909722
CEC-BBOB-2015-F109	0.088973	0.000583	0.733730	0.104110	N/A
CEC-BBOB-2015-F110	N/A	0.000440	0.733730	0.001008	0.031209

parameters. The results can be improved with increasing the number of solutions.

In summary, the preceding subsections employed a set of test functions with diverse characteristics to confidently benchmark and confirm the performance of SSA qualitatively and quantitatively. It was observed that the SSA algorithm shows high exploration and local optima avoidance. This originates from the fact that the artificial salps in SSA tend to interact with each other, so they do not gravitate towards a local solution easily. The salp chain simulated allows the SSA to explore the search space and gradually move towards the global optimum. Another finding of this section was the superior exploitation of the SSA algorithm. This is due to the storing the best solution obtained so far in each iteration and the tendency of salps towards it. The connections between the salps also pull the swarm towards the global optimum. The entire swarm converges towards the optimum proportional to the iteration number due to the utilized adaptive mechanism. In addition, it was observed and confirmed that SSA effectively balances exploration and exploitation. This is again because of the adaptive operator integrated in SSA, in which the early stages of optimization are dedicated to exploration, whereas the exploitation is emphasised in the final iterations.

The results, analysis, and finding of the subsection make the SSA algorithm potentially capable of solving real-world problems with unknown search spaces. In a real search space, the location of the global optimum is unknown. Therefore, the balance of exploration and exploitation highly increases the chance of determining the global optimum. The superiority of SSA was mainly due to this reason which originated from the adaptive mechanism. SSA outperforms other swarm-based techniques (e.g. PSO, FA, BA, etc.) due to higher exploration during optimization. Such techniques show less exploration compared to evolutionary algorithms equipped with crossover operators. However, the SSA algorithm prevents solutions from a rapid convergence towards a local solution. On the other hand, SSA highly promotes exploitation using the c_1 parameter in the final steps of optimization. This assisted SSA to outperform evolutionary algorithms (e.g. GSA and GA) in terms of the accuracy of results.

Although the above results show the effectiveness of SSA in solving a wide range of challenging problems, a number of real

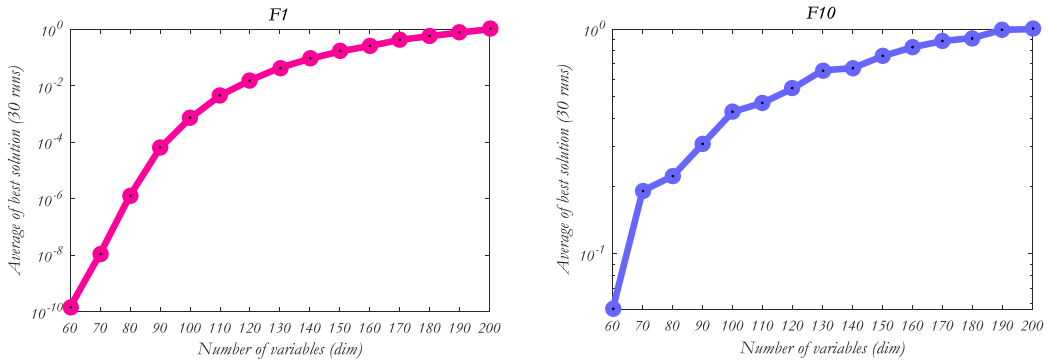

Fig. 11. Results of SSA on F1 and F10 with a varied number of parameters.

Table 7
Results of the multi-objective algorithms (using IGD) on the ZDT test functions employed.

Algorithm	ZDT1					ZDT2				
	Ave	*Std.*	*Median*	*Best*	*Worst*	*Ave*	*Std.*	*Median*	*Best*	*Worst*
MSSA	0.00286	0.000841427	0.0025	0.0023	0.0043	0.0037	0.00130958	0.0044	0.0015	0.0047
MOPSO	0.00422	0.003103	0.0037	0.0015	0.0101	0.00156	0.000174	0.0017	0.0013	0.0017
NSGA-II	0.05988	0.005436	0.0574	0.0546	0.0702	0.13972	0.026263	0.1258	0.1148	0.1834
Algorithm	ZDT3					ZDT1 with linear front				
	Ave	*Std.*	*Median*	*Best*	*Worst*	*Ave*	*Std.*	*Median*	*Best*	*Worst*
MSSA	0.02986	0.000898888	0.0296	0.0291	0.0314	0.0033	0.000731437	0.0034	0.0025	0.0041
MOPSO	0.03782	0.006297	0.0362	0.0308	0.0497	0.00922	0.005531	0.0098	0.0012	0.0165
NSGA-II	0.04166	0.008073	0.0403	0.0315	0.0557	0.08274	0.005422	0.0804	0.0773	0.0924
Algorithm	ZDT2 with 3 objectives									
	Ave	*Std.*	*Median*	*Best*	*Worst*					
MSSA	0.00786	0.001583667	0.0078	0.0059	0.0098					
MOPSO	0.02032	0.001278	0.0203	0.0189	0.0225					
NSGA-II	0.0626	0.017888	0.0584	0.0371	0.0847					

problems is solved in the next section to demonstrate the performance of the SSA algorithm in practice. Note that the real case studies are constrained. Therefore, it is essential to equip SSA with a constraint handling mechanism. Since investigation and finding a suitable constraint handling technique for SSA is outside the scope of his work, the simplest constraint handling method is employed, in which a death (barrier) penalty function penalizes the search agents with a very large objective value in case of violation of any constraints at any level. This technique will be used in MSSA as well. This causes ignoring the infeasible search agents automatically by the SSA algorithm.

It is worth mentioning here that SSA is able to solve problems (including NP hard) with continuous variables only. It should be equipped with suitable operators to solve binary problems such as feature selection or clustering. Since this algorithm considers problems as a black box and according to the NFL theorem, it can be applied to any type of problem subject to proper formulation and appropriate modifications.

4.5. Results of MSSA and discussion

4.5.1. ZDT test problems

This subsection investigates the efficacy of the MSSA proposed in this work experimentally. Similarly to the single-objective optimization, there are different standard teste suites in the literature. A set of five challenging test functions called ZDT proposed by Zitzler et al. [90] is employed here to benchmark the performance of MSSA. The mathematical models of these benchmark functions are presented in the Appendix B. It is worth noting here that the first three benchmark functions are ZDT1 to ZDT 3 functions. Also, the modified ZDT1 and ZDT2 with linear and three-objective fronts are taken from [71] to have different test functions.

For results verification, the two most well-regarded algorithms in the literature of multi-objective optimization are selected: MOPSO and NSGA-II. Due to the difficulty of multi-objective test functions compared to single-objective ones, more search agents (60) and a larger maximum iteration (1000) are employed. Each algorithm is run 30 times and quantitative results are calculated using IGD and presented in Table 7. Also, the best Pareto optimal front obtained are illustrated in Figs. 12–16. Note that the maximum archive size for both MSSA and MOPSO are set to 100.

Table 7 shows that the MSSA algorithm significantly outperforms both MOPSO and NSGA-II on the majority of ZDT test problems. Inspecting Pareto optimal fronts obtained in Fig. 12, it may be seen that MSSA and MOPSO show a better convergence than NSGA-II. The distribution of the solutions is nearly uniform for both MSSA and MOPSO algorithms, which shows the high coverage of these algorithms. There is a gap in the middle of the Pareto optimal front obtained by the NSGA-II algorithm, which negatively impacts the coverage of this algorithm. ZDT1 has a concave-shaped Pareto optimal front and is always challenging for aggregation-based methods. However, these results show that MSSA is able to efficiently approximate the true front of this function with a very high convergence and coverage. Between MSSA and MOPSO, Fig. 12 shows that the convergence of MSSA is better and the coverage is very competitive.

By contrast, the Pareto optima front of ZDT2 function is convex. Therefore, the convergence and coverage of algorithms on a different Pareto optimal front can be benchmarked. The Pareto optimal solutions obtained in Fig. 13 show that MSSA and MOPSO again

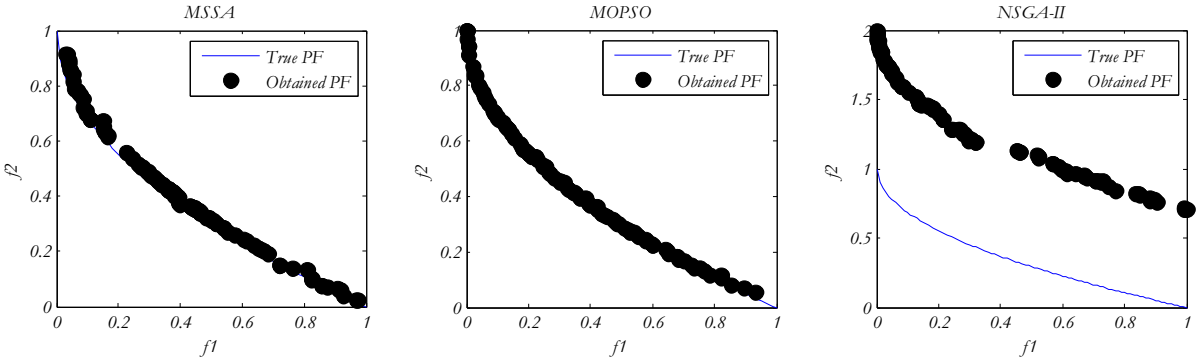

Fig. 12. Best Pareto front determined by MSSA, MOPSO, and NSGA-II on ZDT1.

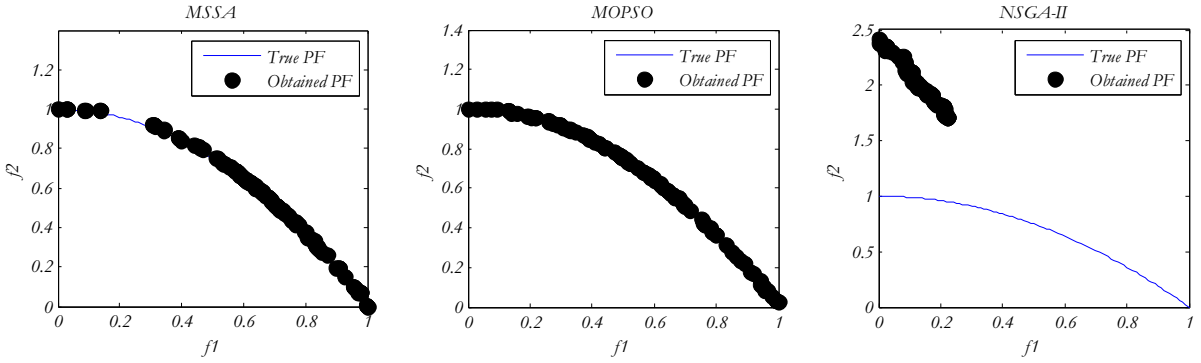

Fig. 13. Best Pareto front determined by MSSA, MOPSO, and NSGA-II on ZDT2.

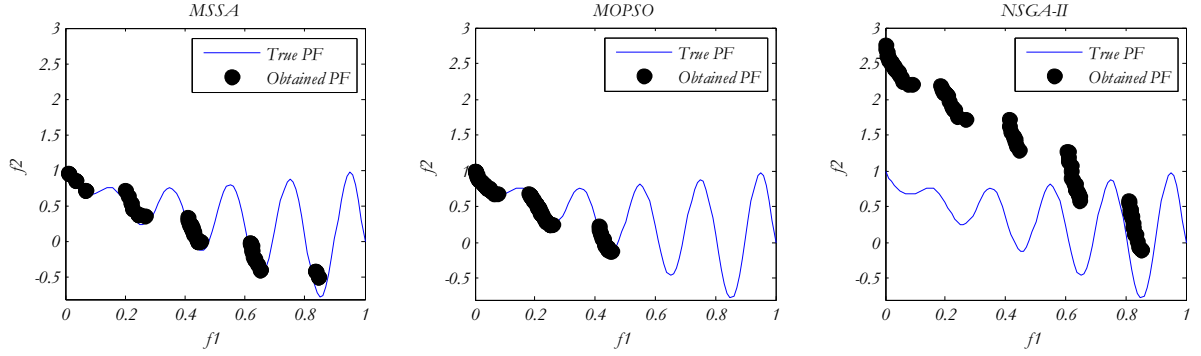

Fig. 14. Best Pareto front determined by MSSA, MOPSO, and NSGA-II on ZDT3.

both outperform NSGA-II algorithm. The convergence and coverage of the NSGA-II is very low. It seems that the convergence of MSSA is better than MOPSO, yet the coverage is slightly lower. There is a gap in the Pareto optimal front obtained by the MSSA algorithm. However, the distribution of solutions is highly uniform in the rest of the front. The results of MSSA on ZDT1 and ZDT show that MSSA is able to approximate concave and convex Pareto optimal front with a reasonable convergence and coverage.

As may be seen in Fig. 14, ZDT3 has a Pareto optimal front with separated regions. These types of fronts are common in real problems and very challenging to be determined by optimization algorithms. This is because an algorithm might be trapped in one of the regions and fail to approximate all the separated regions. Comparing the results on ZDT3 and those on the previous ZDT test functions, a similar pattern can be seen in the Pareto optimal fronts obtained, in which NSGA-II shows the worst results. Despite the high coverage of this algorithm, the convergence is poor. Also, Pareto optimal solutions obtained on some of the separated regions are very far from the true front. Considering the fronts of MSSA and MOPSO, it is evident that MSSA outperforms MOPSO in terms of convergence and specially coverage. These results demonstrate that MSSA is able to effectively find all the separated regions of a Pareto optimal front with high distribution on each region.

As the fourth case study, ZDT1 is equipped with a linear front and able to benchmark the coverage of the algorithms clearly. The results in Fig. 15 show that again NSGA-II failed to converge towards the true Pareto optimal front. All of the Pareto optimal solutions obtained are far from the true front. Similarly to ZDT3, the coverage of NSGA-II is reasonable despite the poor convergence. Inspecting the fronts of MSSA and MOPSO in Fig. 15, it can be seen that MSSA provides superior results in terms of both convergence and coverage. The results of MOPSO are very competitive and of course better than NSGA-II. These results demonstrate that MSSA efficiently drives salps towards different regions of true Pareto optimal front.

The first four case studies have two objectives and the above results prove that MSSA is very effective in solving such problems. However, some of problems have three objectives. The last case

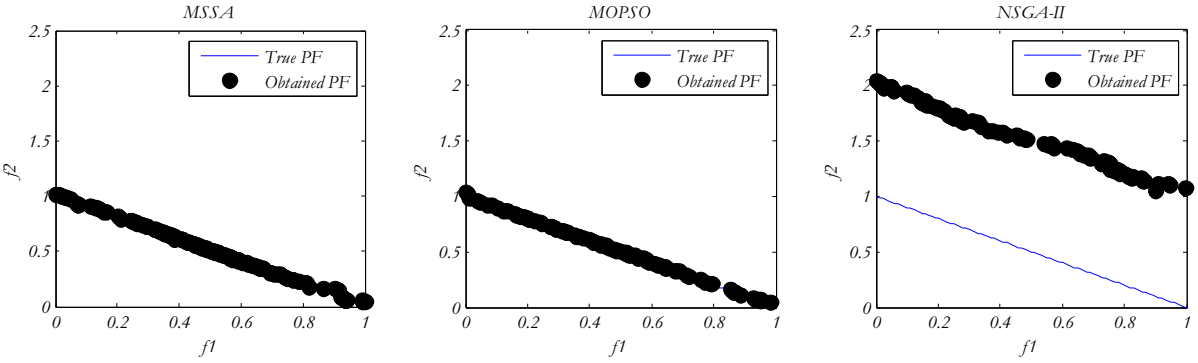

Fig. 15. Best Pareto front determined by MSSA, MOPSO, and NSGA-II on ZDT1 with linear front.

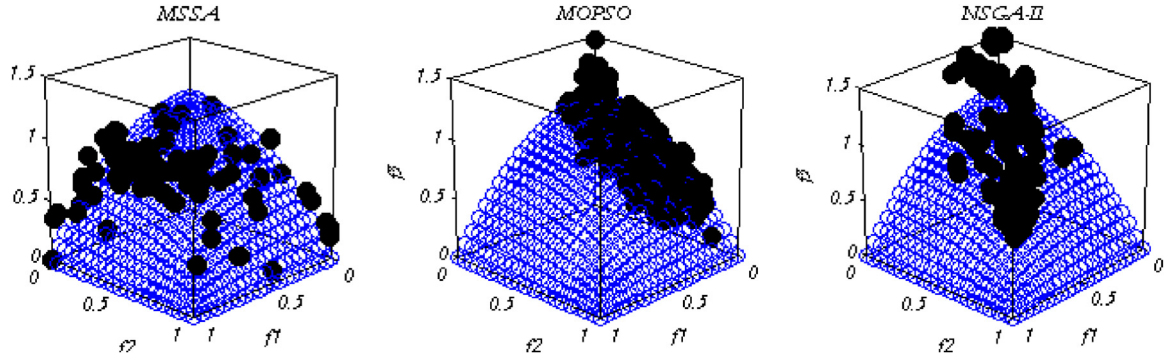

Fig. 16. Best optimal front determined by MSSA, MOPSO, and NSGA-II on ZDT2 with 3 objectives.

study is the extended version of ZDT2 with 3 objectives that provides a very challenging tri-objective test bed for the algorithms. Pareto optimal solutions obtained in Fig. 16 show that similarly to other test functions, NSGA-II presents the worst convergence and coverage. By contrast, the MSSA algorithm provides the best convergence and coverage. The results of MOPSO show that this algorithm only converge towards one region of the front and the coverage is low. These results indicate that the MSSA algorithm is capable of approximating the true Pareto optimal fronts of tri-objective optimization problems as well.

4.5.2. CEC2009 test functions

The CEC2009 test suite [91] is considered as one of the most challenging benchmark sets in the literature and includes highly biased, rotated, shifted, hybridized, and composite test functions. The Pareto optimal front of these functions are of different shapes and continuity. This sub-section applies the MSSA algorithm to these test problems and compares the results with MOPSO and MOEA/D. The results are given in Table 8. Note that to challenge the MSSA algorithm further, it is compared to MOPSO and MOEA/D on this test suite.

Inspecting the results of this table, it is evident that MSSA shows the best results on eight test functions and provide very competitive results when solving the rest of CEC2009 test functions. These results confirm the performance of the proposed MSSA algorithm when solving challenging test functions with difficulties similar to those in the search space of real-world problems.

Taken together, the main advantages of the proposed MSSA algorithm in comparison with MOEA/D and MOPSO are high convergence and coverage. Superior convergence is due to the leading solution selection, in which one of the best non-dominated solutions always update the position of others. Another advantage is the high coverage of the MSSA algorithm, which is because of both archive maintenance mechanism and the selection of leading solution. Since the solutions are always discarded from most populated segments and leaders are chosen from the least populated segments of the archive, MSSA improves the diversity and coverage of solutions across all objectives. Despite these benefits, MSSA is supposed to be applied to problems with three and maximum four objectives. As a Pareto dominance-based algorithm, MSSA becomes less effective proportional to the number of objectives. This is due to the fact that in problems with more than four objectives, a large number of solutions are non-dominated, so the archive become full quickly. Therefore, the MSSA algorithm is suitable for solving problems with less than four objectives. In addition, this algorithm is suitable only for problems with continuous variables and requires legit modifications to be used in problems with binary variables.

The results proved that MSSA can be very effective for solving optimization problems with multiple objectives. The MSSA algorithm showed high convergence and coverage. The superior convergence of MSSA is due to the updating position of salps around the best non-dominated solutions obtained so far. The salp chain has always tendency towards the best solutions. Also, the high convergence originates from the adaptive mechanism which accelerates the movements of salps toward the best non-dominated solutions obtained so far in the repository. The high coverage of MSSA is because of the repository maintenance and leading solution selection mechanisms. When the repository becomes full, non-dominated solutions in populated regions are discarded by MSSA, which results in improving the distribution of solutions along the entire front. The procedure of selecting the leading salps also emphasizes coverage because it selects solutions from the least populated regions to be explored and exploited by the salp chain. It is worth mentioning here that since the updating mechanism of the leading and follower salps in MSSA are identical to those of SSA, MSSA inherits high exploration, local solutions avoidance, exploitation, and fast convergence rate from this algorithm. Therefore, the

Table 8
Statistical results for IGD on UF1 to UF10 in the CEC2009 test suite.

IGD	UF1 (bi-objective)			UF2 (bi-objective)			UF3 (bi-objective)		
	MSSA	MOPSO	MOEA/D	MSSA	MOPSO	MOEA/D	MSSA	MOPSO	MOEA/D
Average	0.1024	0.1370	0.1871	0.0576	0.0604	0.1223	0.2628	0.3139	0.2886
Median	0.1026	0.1317	0.1828	0.0580	0.0483	0.1201	0.2424	0.3080	0.2892
STD. Dev.	0.0062	0.0440	0.0507	0.0048	0.0276	0.0107	0.0727	0.0447	0.0159
Worst	0.1093	0.2278	0.2464	0.0657	0.1305	0.1436	0.4005	0.3777	0.3129
Best	0.0897	0.0899	0.1265	0.0479	0.0369	0.1048	0.1711	0.2564	0.2634
IGD	*UF4 (bi-objective)*			*UF5 (bi-objective)*			*UF6 (bi-objective)*		
	MSSA	MOPSO	MOEA/D	MSSA	MOPSO	MOEA/D	MSSA	MOPSO	MOEA/D
Average	0.0902	0.1360	0.0681	0.6659	2.2023	1.2914	0.1903	0.6475	0.6881
Median	0.0891	0.1343	0.0684	0.6931	2.1257	1.3376	0.1962	0.5507	0.6984
STD. Dev.	0.0040	0.0073	0.0021	0.0986	0.5530	0.1348	0.0457	0.2661	0.0553
Worst	0.0984	0.1518	0.0703	0.7914	3.0383	1.4674	0.2666	1.2428	0.7401
Best	0.0855	0.1273	0.0646	0.4495	1.4647	1.1230	0.1163	0.3793	0.5523

IGD	UF7 (bi-objective)			UF8 (tri-objective)		UF9 (tri-objective)		UF10 (tri-objective)	
	MSSA	MOPSO	MOEA/D	MSSA	MOPSO	MSSA	MOPSO	MSSA	MOPSO
Average	0.0690	0.3539	0.4552	0.2743	0.5367	0.4441	0.4885	0.9769	1.6371
Median	0.0686	0.3873	0.4376	0.2655	0.5364	0.4222	0.4145	0.9190	1.5916
STD. Dev.	0.0059	0.2044	0.1898	0.0447	0.1825	0.1084	0.1444	0.2189	0.2987
Worst	0.0796	0.6151	0.6770	0.3794	0.7963	0.6422	0.7221	1.3142	2.1622
Best	0.0610	0.0540	0.0290	0.2249	0.2453	0.2849	0.3335	0.6082	1.2200

Table 9
Comparison results of the three-bar truss design problem.

Algorithm	Optimal values for variables		Optimal weight
	x_1	x_2	
SSA	0.788665414258065	0.408275784444547	263.8958434
DEDS [93]	0.78867513	0.40824828	263.8958434
PSO-DE [94]	0.7886751	0.4082482	263.8958433
MBA [47]	0.7885650	0.4085597	263.8958522
Ray and Sain [95]	0.795	0.395	264.3
Tsa [96]	0.788	0.408	263.68
CS [92]	0.78867	0.40902	263.9716

MSSA algorithm can avoid local fronts and converge towards the true Pareto optimal front.

The results, findings, and discussions of this section prove that the MSSA algorithm is potentially able to solve multi-objective optimization problems with unknown search spaces. To experimentally prove this, the next section employs MSSA to solve a real-world multi-objective problem.

5. Real-world applications

This sections applies the SSA and MSSA algorithms proposed in this work to several real-world problems. SSA is employed to solve five classical engineering design problems and airfoil design for aero vehicles. MSSA is required to solve a propeller design problem for marine vehicles. The last two problems are computationally expensive and each function evaluation might take up to five minutes. Also, there are many constraints that should not be violated by the optimal solution(s) obtained.

5.1. Three-bar truss design problem

This classical engineering problem is to design a truss with three bars to minimize its weight. This problem has a highly constrained search space [47,92]. The structural parameters in this problem are shown in Fig. 17.

The results of SSA when solving this problem are shown in Table 9. It can be seen that this algorithm is competitive compared to conventional and stochastic optimization techniques in the literature.

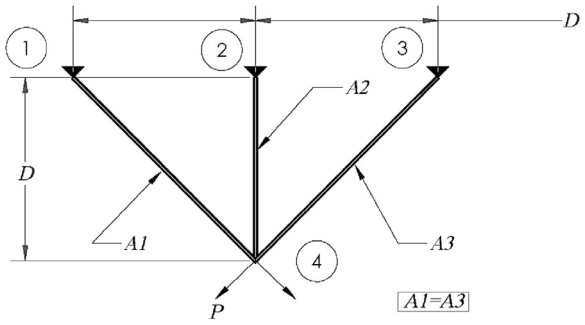

Fig. 17. Three-bar truss design problem.

5.2. Welded beam design problem

As shown in Fig. 18, the objective in this classical problem is to design a welded beam with minimum fabrication cost [97].

This problem is solved by SSA, and the results are compared with several techniques in the literature [98–103]. Table 10 presents the results and shows that SSA finds the minimum optimal cost.

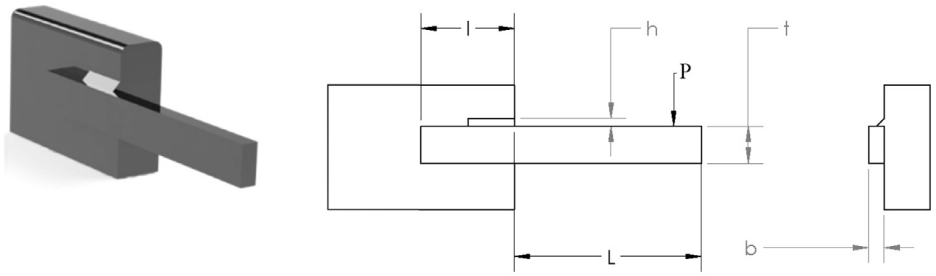

Fig. 18. Design parameters of the welded beam design problem.

Table 10
Comparison results of the welded beam design problem.

Algorithm	Optimal values for variables				Optimal cost
	h	*l*	*t*	*b*	
SSA	0.2057	3.4714	9.0366	0.2057	1.72491
GSA	0.182129	3.856979	10.0000	0.202376	1.87995
CPSO [104]	0.202369	3.544214	9.048210	0.205723	1.73148
GA [98]	0.1829	4.0483	9.3666	0.2059	1.82420
GA [100]	0.2489	6.1730	8.1789	0.2533	2.43312
Coello [105]	0.208800	3.420500	8.997500	0.2100	1.74831
Coello and Montes [106]	0.205986	3.471328	9.020224	0.206480	1.72822
Siddall [107]	0.2444	6.2189	8.2915	0.2444	2.38154
Ragsdell [103]	0.2455	6.1960	8.2730	0.2455	2.38594
Random [103]	0.4575	4.7313	5.0853	0.6600	4.11856
Simplex [103]	0.2792	5.6256	7.7512	0.2796	2.53073
David [103]	0.2434	6.2552	8.2915	0.2444	2.38411
APPROX [103]	0.2444	6.2189	8.2915	0.2444	2.38154

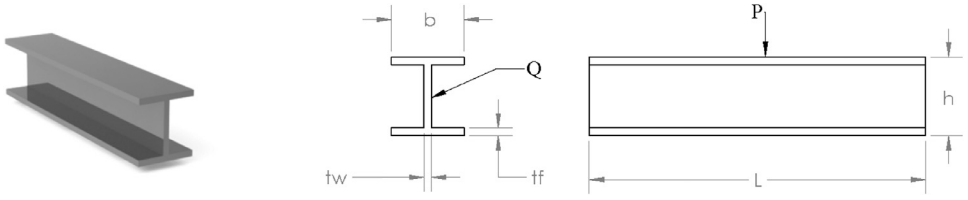

Fig. 19. I-beam design problem.

Table 11
Comparison results for I-beam design problem.

Algorithm	Optimal values for variables				Optimum vertical deflection
	b	*h*	t_w	t_f	
SSA	50	80	1.76470587	5.0000	0.0066259581
ARSM [108]	37.05	80	1.71	2.31	0.0157
IARSM [108]	48.42	79.99	0.90	2.40	0.131
CS [92]	50	80	0.9	2.321675	0.0130747
SOS [109]	50	80	0.9	2.32179	0.0130741

5.3. I-beam design problem

The I-beam problem deals with designing an I-beam with four structural parameters to minimize the vertical deflection of the beam (see Fig. 19).

This problem is solved by SSA in Table 11. It is evident that SSA significantly outperforms other techniques when solving this problem.

5.4. Cantilever beam design problem

Despite the similarity of this problem to the preceding one, the objective is to minimize the weight of a cantilever beam Cantilever beam consists of five hollow blocks as shown in Fig. 20. The optimal results obtained by SSA and similar techniques in the literature are given in Table 12. It may be seen that the proposed SSA algorithm outperforms the majority of techniques and shows very competitive results compared to SOS.

5.5. Tension/compression spring design

The last classical engineering problem is to design a compression spring with three parameters as illustrated in Fig. 21 [106,111,112]. To perform a fair comparison with literature, a penalty function is used in a similar manner to [113]. The results in

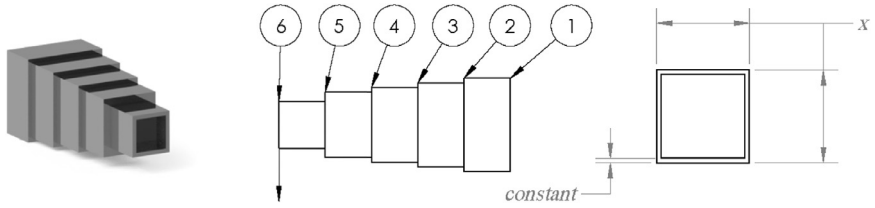

Fig. 20. Cantilever beam design problem.

Table 12
Comparison results for cantilever design problem.

Algorithm	Optimal values for variables					Optimum weight
	x_1	x_2	x_3	x_4	x_5	
SSA	6.015134526133134	5.309304676055819	4.495006716308508	3.501426286300545	2.152787908005768	1.339956391038955
SOS [109]	6.01878	5.30344	4.49587	3.49896	2.15564	1.33996
MMA [110]	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
GCA_I [110]	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
GCA_II [110]	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
CS [92]	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999

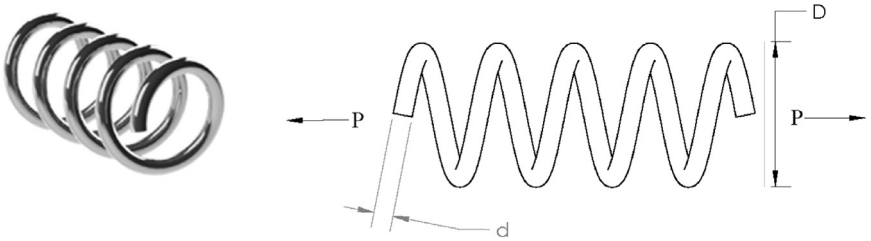

Fig. 21. Schematic of the compression spring.

Table 13
Comparative results for the tension/compression spring design problem.

Algorithm	Optimum variables			Optimum weight
	d	D	N	
SSA	0.051207	0.345215	12.004032	0.0126763
GSA	0.050276	0.323680	13.525410	0.0127022
PSO [104]	0.051728	0.357644	11.244543	0.0126747
ES [114]	0.051989	0.363965	10.890522	0.0126810
GA (Coello) [115]	0.051480	0.351661	11.632201	0.0127048
RO [116]	0.051370	0.349096	11.76279	0.0126788
Improved HS [117]	0.051154	0.349871	12.076432	0.0126706
DE [118]	0.051609	0.354714	11.410831	0.0126702
Mathematical optimization	0.053396	0.399180	9.1854000	0.0127303
Constraint correction	0.050000	0.315900	14.250000	0.0128334

Table 13 show the merits of proposed SSA in solving this problem as well.

5.6. Two-dimensional airfoil design using SSA

In the airfoil design problem, there are two objectives: minimizing drag versus maximizing lift. In fact, lift and drag are two forces applied to an airplane as can be seen in Fig. 22. Engines are the main sources of propulsion in an aircraft, which provides thrust as the main force. Depending on the shape of aircraft's body and wings, the thrust is converted to both lift and drag. The force lift is in opposite direction of weight, so it causes flying when greater than the weigh force. Despite the low density of air, the body and wing shape of a moving airplane also result is drag.

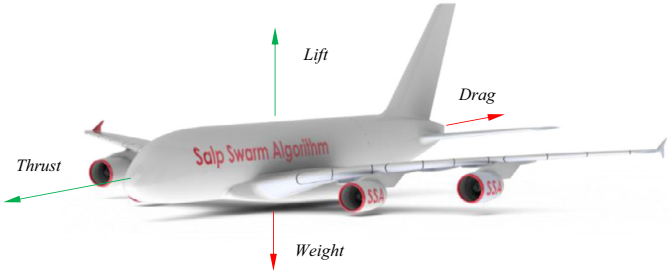

Fig. 22. Different forces applied to an airplane.

The shape of a wing plays an essential role in the performance of an aircraft mostly in terms of lift and drag. A suitable design

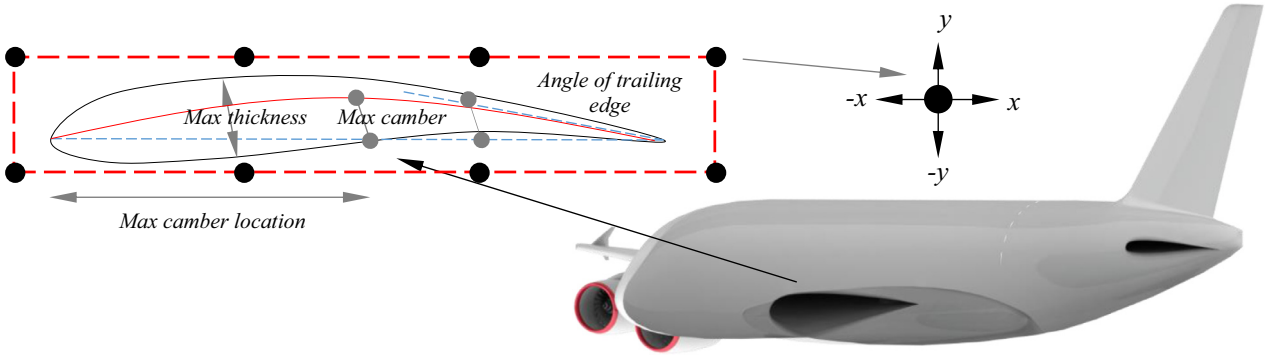

Fig. 23. Cross section of a real wing with a 2D NACA airfoil [121] and B-spline for defining the shape of airfoil.

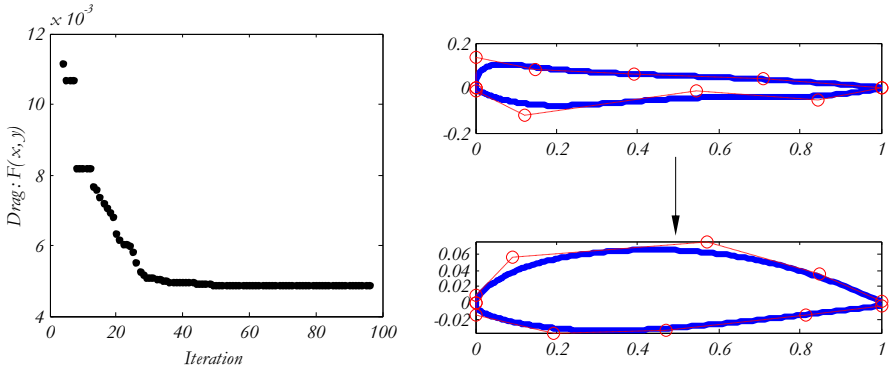

Fig. 24. Convergence curve of the SSA on the airfoil design problem, initial airfoil, and optimized airfoil.

for airfoil can save fuel significantly as well. It cannot be said if lift or drag forces are better because both of them are desirable in different phases of fly. The lift force is desirable and drag is undesirable in take-off and ascend. The higher the lift, the faster ascend and take off. However, the drag force is desirable and lift is undesirable when landing or descending. The higher drag, the faster ascend and landing.

Maximizing lift and minimizing drag are two important objectives in airfoil design. Since the main focus of this section is to demonstrate the applicability of SSA, one of these objectives is optimized. In order to solve this problem with the SSA algorithm, it has to be formulated as an objective function. Problem formulation includes identifying objective function, structural parameters, and constraints. The main objective is to minimize drag in this subsection. The freeware XFoil [119] is used for calculating drag.

A real aircraft wing has many parameters: position of engines, internal frames, leading flaps, trailing flaps, fuel tanks, leading edge, trailing edge, and the shape airfoil. For the sake of simplicity, the shape of airfoil is only optimized in this work. The curvature of an airfoil can be defined with different methods of which b-spline [120] is chosen in this work. Fig. 23 shows how this method utilizes controlling points to define a smooth shape for the airfoil. The model employed can handle up to 8 points. It is considered one of the leading points is fixed to have a reference point, so there are 7 points to be moved in total. As Fig. 23 shows, the displacements of points are along positive/negative x and y directions, so the total number of parameters for optimization is equal to 14.

Airfoil design is a CFD problem and accompanied with a large number of constraints. Since investigating these constraints is out of the scope of the current work, they are not discussion in this section. After all, the problem of airfoil design can be formulated as follows:

$$\begin{aligned} &\textit{Minimize}: \quad F(\vec{x},\ \vec{y}) = C_d(\vec{x},\ \vec{y}) \\ &\textit{Subject to}: \quad -1 \leq \vec{x}, \vec{y} \leq 1, \quad \textit{satisfaction of CO set} \end{aligned} \tag{5.1}$$

where $\vec{x} = \{x_1,\ x_2, \ldots, x_7\}$, $\vec{y} = \{y_1, y_2,\ \ldots, y_7\}$, CO includes many constraints such as minimum of thickness, maximum of thickness, etc.

It should be noted here that CFD problems have mostly dominated infeasible regions, which make the death penalty functions very inefficient. To be able to solve this problem using SSA, the following penalty functions has been employed:

$$F(\vec{x},\ \vec{y}) = F(\vec{x},\ \vec{y}) + p\sum_{i=1}^{3} P_i \tag{5.2}$$

where p is a constant and P_i is the violation size on the ith constraint in the CO set in Eq. (5.1).

For finding an optimal shape for the airfoil employed in this subsection, a salp chain with 30 salps are utilized. The rest of the controlling parameters of SSA is identical to those of the preceding sections except the maximum number of generation, which is set to 1000. The stopping criterion is to reach either a minimum drag or maximum number of iterations. The problem is solved four times and the best results are illustrated in Fig. 24. To see how the SSA solves this problem, the initial random design, the final optimal design found, and convergence curve are also shown in this figure.

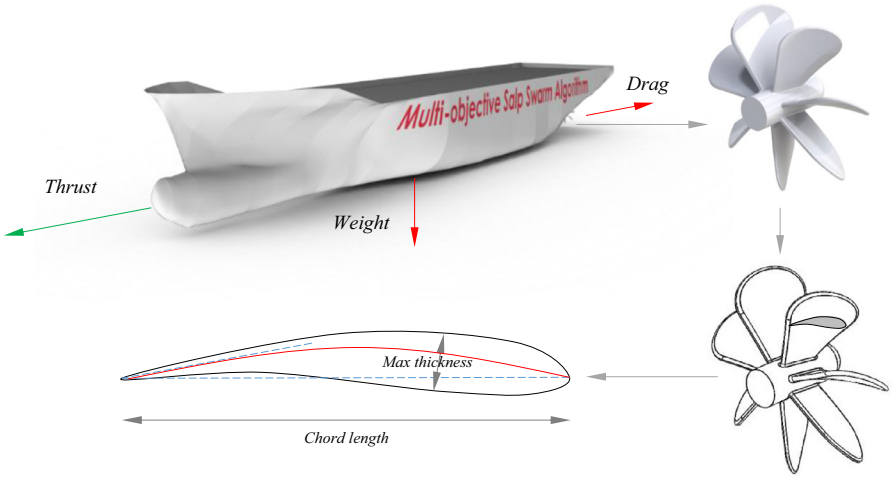

Fig. 25. Cross section of a real propeller and a 2D NACA airfoil.

Inspecting the results in Fig. 24, it may be observed that SSA successfully minimizes drag of the airfoil over the course of iterations. There are some iterations with no improvement in the solution. This is due to the extremely difficult search space of this problem and the many infeasible solutions found during optimization. However, it seems that the SSA managed to find infeasible and better solutions after each non-improving period (for instance at the end of nearly iterations 9 and 25). The optimal airfoil obtained is also very smooth, showing the ability of SSA in solving this problem. These results highly demonstrate that the SSA is able to approximate the global optima of real-world single-objective problems with challenging and unknown search spaces.

5.7. Marine propeller design using MSSA

The problem investigated in this subsection is another CFD problem. In fact, the shape of a marine propeller is optimized. The main two objectives in the propeller design problem are: maximizing efficiency and minimizing cavitation [122]. The engine(s) of a ship rotate a shaft attached to a propeller. The shafts rotate a propeller, which creates trust for the vehicle. The efficiency is the amount of engine power that is eventually converted to thrust. Due the high density of water and frictions involved in the motor, however, thrust without loss of energy cannot be achieved [123].

In addition to thrust, a propeller creates cavitation [124]. When a propeller is rotated in water, it swirls water at high speed. Accelerating water and passing through blades reduce the pressure of water. This results in forming bubbles that collapse and cause strong local shockwaves. These tiny shockwaves erode the surface of a propeller and reduce its life span. Cavitation is undesirable and an optimizer is better to find a design with the lowest cavitation.

The shape of a propeller plays a significant role in determining efficiency and cavitation. A propeller is made of several similar blades grouped around the shaft. To design a propeller, there are many structural parameters in a propeller: number of blades, length of blade, thickness of shaft, length of shaft, and the shape of blade. Undoubtedly, the blade's shape is the most important components in defining efficiency and cavitation.

There are different approaches for designing a blade. Similarly to airfoil design in the previous section, b-spline can be employed. This approach is usually used when a designer does not want to utilize a standard airfoil. To have a different airfoil design compared to the previous subsection, a standard NACA aoirfoil is chosen in this subsection. As may be seen in Fig. 25, the main structural parameters in the NACA airfoil employed are the maximum thickness and chord length. Multiple numbers of this airfoil along the length of a blade define its final shape. In this work the blade is divided to 10 cross sections each of which is defined by the NACA airfoil shown in Fig. 25.

Blades' airfoil design is a CFD problem and accompanied with a large number of constraints. Since investigating these constraints is out of the scope of the current work, they are not discussed in this section. After all, the problem of propeller design can be formulated as follows:

$$\begin{aligned} &\textit{Maximize}: \quad F_1\left(\vec{T},\ \vec{C}\right) = \eta\left(\vec{T},\vec{C}\right) \\ &\textit{Minimize}: \quad F_2\left(\vec{T},\ \vec{C}\right) = \mathrm{V_c}\left(\vec{T},\vec{C}\right) \\ &\textit{Subject to}: \quad \text{thrust} = 4000, \quad \textit{satisfaction of all constraints} \end{aligned} \tag{5.1a}$$

where $\vec{T}$ is a vector that stores the thickness of all airfoils along the blade ($\vec{T} = \{T_1,\ T_2, \ldots, T_{10}\}$), $\vec{C}$ is a vector that stores the chord length of all airfoils along the blade ($\vec{C} = \{C_1, C_2,\ \ \ldots, C_{10}\}$).

For determining the true Pareto optimal front of this problem, a salp chain with 200 salps are employed. The salps are allowed to find the Pareto optima solutions over 300 iterations. The maximum archive size is set to 100 as well. The rest of the controlling parameters of the MSSA algorithm is identical to previous experiments done with SSA and MSSA. Note that the freeware Openprop [125] is utilized for calculating the objectives. The experiment is performed four times and the best Pareto optimal solution obtained is illustrated in Fig. 26. In addition to the best Pareto optimal front, search history of salp chain and some of the design found are illustrated in this figure. Note that the y axis shows negative cavitation, so best solutions are towards top-right corner of the figure.

As per the results of Fig. 26, it may be seen that the Pareto optimal solutions obtained are well distributed along both objectives. The true Pareto optimal front for this question is unknown, so it cannot be said how close these solutions are to the true Pareto optimal solutions. However, it is evident that the MSSA managed to find solutions with high efficiency and low cavitation. The search history also shows how the salp chain gradually moves towards the Pareto optimal front and spread the solutions across both ob-

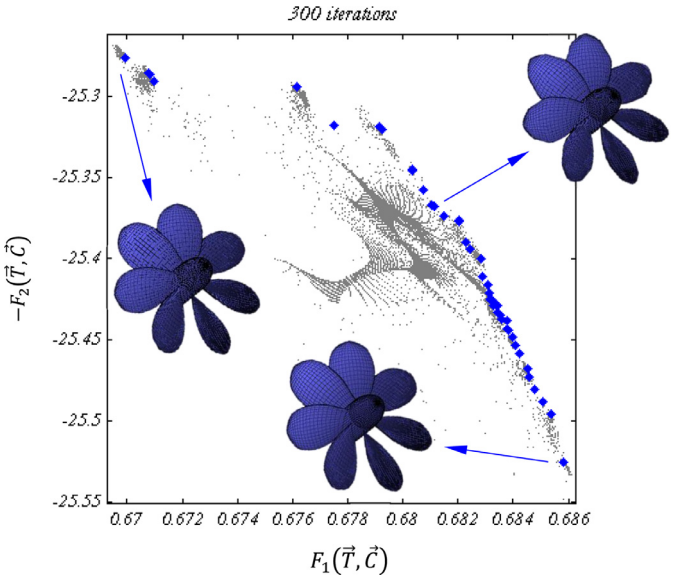

Fig. 26. Pareto optimal front obtained, search history, and some of the designs.

jectives. These results highly demonstrate that the model of salp swarm proposed in this work and operators employed for solving multi-objective problems are very efficient in terms of finding an accurate approximation and highly distributed Pareto optimal solutions for challenging multi-objective problems with unknown search spaces.

6. Conclusion

This paper proposed a novel nature-inspired technique for solving optimization problems. The swarming behaviour of salps (salp chain) was the main inspiration of this paper. Two mathematical model were proposed to update the position of leading and follower salps. Swarm simulation in 2D and 3D space showed that the models proposed are able to search around both stationary and mobile food sources. After swarm simulation, the SSA and MSSA algorithms were designed. In SSA, the best solutions obtained so far was considered as the main food source to be chased by the salp chain. An adaptive mechanism was integrated to SSA to balance exploration and exploitation. For the MSSA algorithm, a repository was designed and employed to store non-dominated solutions obtained so far. Solutions were removed from population regions in case of a full repository and the food source was chosen from the non-dominated solutions in the least populated areas.

In order to prove the efficacy of the algorithms proposed a series of experiments was conducted. First, several qualitative metrics were employed: search history, trajectory, average fitness, and convergence curve. SSA was applied to a set of benchmark functions. It was observed and can be concluded that SSA is able to explore the most promising regions of the search space, move salps abruptly in the initial steps of iterations, move salps gradually in the final stages of iterations, improve the average fitness of all salps, and improve that best solution found so far over the course of optimization.

Also, a set of high-dimensional test functions including unimodal, multi-modal, and composite were solved by the SSA algorithm to prove its performance in solving problems with a large number of variables and different characteristics. The results were compared with a variety of well-known and recent algorithms using a statistical test. It was observed and may be concluded that SSA is capable of determining the global optima for most of the unimodal, multi-modal, and composite benchmark functions and outperform the current optimization techniques in the literature in a statistically significant manner.

For proving the performance of MSSA algorithm, a set of well-known multi-objective test functions was employed. The results were compared with MOPSO and NSGA-II as the best two algorithms proposed so far in the literature. As per the results and finding, it can be concluded that the MSSA algorithm can approximate the true Pareto optimal front with different shapes and difficulties conveniently. It is also concluded that the MSSA benefits from a reasonable convergence and coverage that allows this algorithm to find very accurate and highly distributed approximation of Pareto optimal solutions across all objectives.

Although the results on test functions testified the potential of the SSA and MSSA algorithms in solving real problems, this work also considered solving two real problems (arifoil design and marine propeller design) to prove the effectiveness of these algorithms in practice. It was demonstrated that the SSA and MSSA are able to find optimal shapes for both problems employed. As per the results on real-world case studies, it can be concluded that SSA and MSSA can solve real-world problems with unknown search spaces.

According to the simulations, results, finding, analyses, discussions, and conclusions, it can be stated that the SSA and MSSA algorithms have merits among the current optimization algorithms in the literature and worth applying to different problems.

This work opens several research directions. Solving single- and multi-objective problems in different fields is recommended. Proposing binary versions of both SSA and MSSA could be valuable contributions as well. The current work briefly touched on constrained optimization using the algorithms proposed. Therefore, investigating the impacts of different constrained handling methods on the performance of SSA and MSSA is recommended.

Appendix A. Single-objective test problems utilised in this work

Table 14, 15, and 16 show the details of the single-objective test functions employed in this work.

Table 14
Unimodal benchmark functions.

Function	Dim	Range	Shift position	f_{min}
$F_1(x)=\sum_{i=1}^{n} x_i^2$	20	[−100,100]	[−30,−30,...,−30]	0
$F_2(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	20	[−10,10]	[−3,−3,...,−3]	0
$F_3(x)=\sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	20	[−100,100]	[−30,−30,...,−30]	0
$F_4(x)=\max_i\{\lvert x_i\rvert,\ 1\le i\le n\}$	20	[−100,100]	[−30,−30,...,−30]	0
$F_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	20	[−30,30]	[−15,−15,...,−15]	0
$F_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	20	[−100,100]	[−750,...,−750]	0
$F_7(x)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	20	[−1.28,1.28]	[−0.25,...,−0.25]	0

Table 15
Multimodal benchmark functions.

Function	Dim	Range	Shift position	f_{min}
$F_8(x)=\sum_{i=1}^{n}-x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$	20	[−500,500]	[−300,..,−300]	−418.9829 × 5
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10cos(2\pi x_i)+10]$	20	[−5.12,5.12]	[−2,−2,..,−2]	0
$F_{10}(x)=-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-exp\left(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i)\right)+20+e$	20	[−32,32]		0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	20	[−600,600]	[−400,..,−400]	0
$F_{12}(x)=\frac{\pi}{n}\left\{10sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$				
$y_i=1+\frac{x_i+1}{4}$	20	[−50,50]	[−30,−30,..,−30]	
$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$				0
$F_{13}(x)=0.1\left\{sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2[1+sin^2(2\pi x_n)]\right\}+\sum_{i=1}^{n}u(x_i,5100,4)$	20	[−50,50]	[−100,..,−100]	0

Table 16
Composite benchmark functions.

Function	Dim	Range	f_{min}
F_{14} (CF1):			
$f_1,f_2,f_3,\ldots,f_{10}$ = Sphere Function	10	[−5,5]	0
$[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$			
$[\lambda_1,\lambda_2,\lambda_3\ldots,\lambda_{10}]=[5/100,\ 5/100,\ 5/100,\ ..,\ 5/100]$			
F_{15} (CF2):			
$f_1, f_2, f_3,\ldots, f_{10}$ = *Griewank's Function*			
$[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$	10	[−5,5]	0
$[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[5/100,\ 5/100,\ 5/100,\ ..,\ 5/100]$			
F_{16} (CF3):			
$f_1, f_2, f_3,\ldots, f_{10}$ = *Griewank's Function*			
$[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$	10	[−5,5]	0
$[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[1,\ 1,\ 1,\ ..,\ 1]$			
f_{17} (CF4):			
f_1, f_2 = Ackley'sFunction			
f_3, f_4 = Rastrigin's Function			
f_5,f_6 = Weierstrass Function	10	[−5,5]	0
f_7, f_8 = Griewank's Function			
f_9,f_{10} = Sphere Function			
$[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$			
$[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[5/32,\ 5/32,\ 1,\ 1,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100,\ 5/100,\ 5/100]$			
f_{18} (CF5):			
f_1, f_2 = Rastrigin's Function			
f_3,f_4 = Weierstrass Function	10	[−5,5]	0
f_5, f_6 = Griewank's Function			
f_7, f_8 = Ackley'sFunction			
f_9,f_{10} = Sphere Function			
$[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$			
$[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[1/5,\ 1/5,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100,\ 5/32,\ 5/32,\ 5/100,\ 5/100]$			
f_{19} (CF6):			
f_1, f_2 = Rastrigin's Function	10	[−5,5]	0
f_3,f_4 = Weierstrass Function			
f_5, f_6 = Griewank's Function			
f_7, f_8 = Ackley'sFunction			
f_9,f_{10} = Sphere Function			
$[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1]$			
$[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[0.1^*1/5,\ 0.2^*1/5,\ 0.3^*5/0.5,\ 0.4^*5/0.5,\ 0.5^*5/100,$			
$0.6^*\ 5/100,\ 0.7^*5/32,\ 0.8^*\ 5/32,\ 0.9^*5/100,\ 1^*5/100]$			

Appendix B. Multi-objective test problems utilised in this work

ZDT1:

$$\text{Minimise}: \quad f_1(x) = x_1 \tag{B.1}$$

$$\text{Minimise}: \quad f_2(x) = g(x) \times h(f_1(x), g(x)) \tag{B.2}$$

$$\text{Where}: \quad G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i \tag{B.3}$$

$$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$$
$$0 \le x_i \le 1, \ 1 \le i \le 30 \tag{B.4}$$

ZDT2:

$$\text{Minimise}: \quad f_1(x) = x_1 \tag{B.5}$$

$$\text{Minimise}: \quad f_2(x) = g(x) \times h(f_1(x), g(x)) \tag{B.6}$$

$$\text{Where}: \quad G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i \tag{B.7}$$

$$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$$
$$0 \le x_i \le 1, \ 1 \le i \le 30 \tag{B.8}$$

Table 17
Bi-objective test problems (CEC2009).

Name	Mathematical formulation
UF1	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2,\ f_2 = 1-\sqrt{x} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$
UF2	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j{}^2,\ f_2 = 1-\sqrt{x} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j{}^2$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$ $y_j = \begin{cases} x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]\cos(6\pi x_1 + \frac{j\pi}{n}) & if\ j \in J_1 \\ x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]\sin(6\pi x_1 + \frac{j\pi}{n}) & if\ j \in J_2 \end{cases}$
UF3	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ $f_2 = \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ J_1 and J_2 are the same as those of $UF1$, $y_j = x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})},\ j = 2,3,\ldots,n$
UF4	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h(y_j),\ f_2 = 1 - x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h(y_j)$ J_1 and J_2 are the same as those of $UF1$, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2,3,\ldots,n,\ h(t) = \frac{\lvert t\rvert}{1+e^{2\lvert t\rvert}}$
UF5	$f_1 = x_1 + \left(\frac{1}{2N} + \epsilon\right)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h(y_i),\ f_1 = 1 - x_1 + (\frac{1}{2N} + \epsilon)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h(y_i)$ J_1 amd J_2 are identical to those of $UF1$, $\epsilon > 0,\ y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2,3,\ldots,n$ $h(t) = 2t^2 - \cos(4\pi t) + 1$
UF6	$f_1 = x_1 + \max\left\{0, 2(\frac{1}{2N} + \epsilon)\sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 1\right)\right)$ $f_2 = 1 - x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \epsilon\right)\sin(2N\pi x_1)\right\}\frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 1\right)\right)$ J_1 amd J_2 are identical to those of $UF1$, $\epsilon > 0,\ y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}),\ j = 2,3,\ldots,n$
UF7	$f_1 = \sqrt[5]{x_1} + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2,\ f_2 = 1 - \sqrt[5]{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2$ J_1 amd J_2 are identical to those of $UF1, \epsilon > 0, y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2,3,\ldots,n$

Table 18
Tri-objective test problems (CEC2009).

Name	Mathematical formulation
UF8	$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n})^2)$ $f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n})^2)$ $f_3 = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n})^2)$ $J_1 = \{j \mid 3 \le j \le n,\ \text{and } j-1 \text{ is a multiplication of } 3\},\ J_2 = \{j \mid 3 \le j \le n,\ \text{and } j-2 \text{ is a multiplication of } 3\},$ $J_3 = \{j \mid 3 \le j \le n,\ \text{and } j \text{ is a multiplication of } 3\},$
UF9	$f_1 = 0.5[\max\{0, (1+\epsilon)(1-4(2x_1-1)^2)\} + 2x_1]x_2 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n})^2)$ $f_2 = 0.5[\max\{0, (1+\epsilon)(1-4(2x_1-1)^2)\} + 2x_1]x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n})^2)$ $f_3 = 1 - x_2 + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n})^2)$ $J_1 = \{j \mid 3 \le j \le n,\ \text{and } j-1 \text{ is a multiplication of } 3\},\ J_2 = \{j \mid 3 \le j \le n,\ \text{and } j-2 \text{ is a multiplication of } 3\},$ $J_3 = \{j \mid 3 \le j \le n,\ \text{and } j \text{ is a multiplication of } 3\},\ \varepsilon = 0.1$
UF10	$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1]$ $f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1]$ $f_3 = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1]$ $J_1 = \{j \mid 3 \le j \le n,\ \text{and } j-1 \text{ is a multiplication of } 3\},\ J_2 = \{j \mid 3 \le j \le n,\ \text{and } j-2 \text{ is a multiplication of } 3\},$ $J_3 = \{j \mid 3 \le j \le n,\ \text{and } j \text{ is a multiplication of } 3\},$

ZDT3:

$$\text{Minimise}: \quad f_1(x) = x_1 \tag{B.9}$$

$$\text{Minimise}: \quad f_2(x) = g(x) \times \ h(f_1(x), g(x)) \tag{B.10}$$

$$\text{Where}: \quad G(x) = 1 + \frac{9}{29}\sum_{i=2}^{N} x_i \tag{B.11}$$

$$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right)\sin\left(10\pi f_1(x)\right)$$
$$0 \le x_i \le 1, \ \ 1 \le i \le 30 \tag{B.12}$$

ZDT1 with linear PF:

$$\text{Minimise}: \quad f_1(x) = x_1 \tag{B.13}$$

$$\text{Minimise}: \quad f_2(x) = g(x) \times \ h(f_1(x), g(x)) \tag{B.14}$$

$$\text{Where}: \quad G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i \tag{B.15}$$

$$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$$
$$0 \le x_i \le 1, \ \ 1 \le i \le 30 \tag{B.16}$$

ZDT2 with three objectives:

$$\text{Minimise}: \quad f_1(x) = x_1 \tag{B.17}$$

$$\text{Minimise}: \quad f_2(x) = x_2 \tag{B.18}$$

$$\text{Minimise}: \quad f_3(x) = g(x) \times \ h(f_1(x), g(x)) \times \ h(f_2(x), g(x)) \tag{B.19}$$

$$\text{Where}: \quad G(x) = 1 + \frac{9}{N-1}\sum_{i=3}^{N} x_i \tag{B.20}$$

$$h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2$$
$$0 \le x_i \le 1, \ \ 1 \le i \le 30 \tag{B.21}$$

The details of CEC2009 multi-objective test problems are given in Table 17 and 18.

References

[1] Bäck T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press; 1996.
[2] Blum C, Li X. Swarm intelligence in optimization. Springer; 2008.
[3] Goldberg DE, Holland JH. Genetic algorithms and machine learning. Mach Learn 1988;3:95–9.
[4] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341–59.
[5] Rechenberg I. Evolution strategy: optimization of technical systems by means of biological evolution, 104. Stuttgart: Fromman-Holzboog; 1973.
[6] Fogel LJ, Owens AJ, Walsh MJ. Artificial intelligence through simulated evolution; 1966.
[7] Yao X, Liu Y, Lin G. Evolutionary programming made faster. Evol Comput IEEE Trans 1999;3:82–102.
[8] Simon D. Biogeography-based optimization. Evol Comput IEEE Trans 2008;12:702–13.
[9] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life; 1991. p. 134–42.
[10] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science; 1995. p. 39–43.
[11] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71.
[12] Yang X-S, Deb S. Cuckoo search via Lévy flights. In: Nature & biologically inspired computing, 2009. NaBIC 2009. world congress on; 2009. p. 210–14.
[13] Yang XS. Firefly algorithm. Eng Optim 2010:221–30.
[14] Yang X-S. A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer; 2010. p. 65–74.
[15] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61.
[16] Kumar V, Chhabra JK, Kumar D. Grey wolf algorithm-based clustering technique. J Intell Syst 2017;26:153–68.
[17] Aswani R, Ghrera S, Chandra S. A novel approach to outlier detection using modified grey wolf optimization and k-nearest neighbors algorithm. Indian J Sci Technol 2016;9.
[18] Kaveh A, Farhoudi N. A new optimization method: dolphin echolocation. Adv Eng Software 2013;59:53–70.
[19] Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Software 2016;95:51–67.
[20] Pan W-T. A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl Based Syst 2012;26:69–74.
[21] Geem ZW, Kim JH, Loganathan G. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60–8.
[22] Kumar V, Chhabra JK, Kumar D. A hybrid approach for data clustering using expectation-maximization and parameter adaptive harmony search algorithm. In: Proceedings of 2015 International Conference on Future Computational Technologies; 2015. p. 61–7.
[23] Glover F. Tabu search-part I. ORSA J Comput 1989;1:190–206.
[24] Davis L. Bit-climbing, representational bias, and test suite design. In: ICGA; 1991. p. 18–23.
[25] Lourenço HR, Martin OC, Stutzle T. Iterated local search; 2001. *arXiv preprint math/0102188*.
[26] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simmulated annealing. Science 1983;220:671–80.
[27] Caporossi G, Hansen P, Mladenović N. Variable neighborhood search. In: Metaheuristics. Springer; 2016. p. 77–98.
[28] Alsheddy A, Voudouris C, Tsang EP, Alhindi A. Guided local search. In: Handbook of heuristics. Springer; 2016. p. 1–37.
[29] Wolpert DH, Macready WG. No free lunch theorems for optimization. Evol Comput IEEE Trans 1997;1:67–82.
[30] Yao X. A review of evolutionary artificial neural networks. Int J Intell Syst 1993;8:539–67.
[31] Coello Coello CA. Constraint-handling using an evolutionary multiobjective optimization technique. Civil Eng Syst 2000;17:319–46.
[32] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Inf Sci 7/10/ 2013;237:82–117.
[33] Coello CAC. Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. Front Comput Sci China 2009;3:18–30.
[34] Ngatchou P, Zarei A, El-Sharkawi M. Pareto multi objective optimization. In: Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on; 2005. p. 84–91.
[35] Zhou A, Qu B-Y, Li H, Zhao S-Z, Suganthan PN, Zhang Q. Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol Comput 2011;1:32–49.
[36] Tan KC, Chiam SC, Mamun A, Goh CK. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. Eur J Oper Res 2009;197:701–13.
[37] Wang G-G, Guo L, Gandomi AH, Hao G-S, Wang H. Chaotic krill herd algorithm. Inf Sci 2014;274:17–34.
[38] Gandomi AH, Alavi AH. Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 2012;17:4831–45.
[39] Wang G-G, Gandomi AH, Alavi AH. Stud krill herd algorithm. Neurocomputing 2014;128:363–70.
[40] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm. Inf Sci 2009;179:2232–48.
[41] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mech 2010;213:267–89 /09/01 2010.
[42] Formato RA. Central force optimization: a new nature inspired computational framework for multidimensional search and optimization. In: Nature inspired cooperative strategies for optimization (NICSO 2007). Springer; 2008. p. 221–38.
[43] Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization. Comput Struct 2012;112–113:283–94.
[44] Rao RV, Savsani VJ, Vakharia D. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 2011;43:303–15.
[45] Dai C, Zhu Y, Chen W. Seeker optimization algorithm. In: Computational intelligence and security. Springer; 2007. p. 167–76.
[46] Moosavian N, Kasaee Roodsari B. Soccer league competition algorithm: a novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evol Comput 2014;17:14–24.
[47] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 2013;13:2592–612.

[48] Branke J, Kaußler T, Schmeck H. Guidance in evolutionary multi-objective optimization. Adv Eng Software 2001;32:499–507.
[49] Das I, Dennis JE. Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 1998;8:631–57.
[50] Kim IY, De Weck O. Adaptive weighted-sum method for bi-objective optimization: pareto front generation. Struct Multidiscip Optim 2005;29:149–58.
[51] Messac A, Mattson CA. Generating well-distributed sets of Pareto points for engineering design using physical programming. Optim Eng 2002;3:431–50.
[52] Parsopoulos KE, Vrahatis MN. Particle swarm optimization method in multiobjective problems. In: Proceedings of the 2002 ACM symposium on Applied computing; 2002. p. 603–7.
[53] Deb K. Advances in evolutionary multi-objective optimization. In: Search based software engineering. Springer; 2012. p. 1–26.
[54] Zhang Q, Li H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. Evol Comput IEEE Trans 2007;11:712–31.
[55] Mezura-Montes E, Reyes-Sierra M, Coello CAC. Multi-objective optimization using differential evolution: a survey of the state-of-the-art. In: Advances in differential evolution. Springer; 2008. p. 173–96.
[56] Kumar V, Chhabra JK, Kumar D. Differential search algorithm for multiobjective problems. Procedia Comput Sci 2015;48:22–8.
[57] Sarker R, Abbass HA. Differential evolution for solving multiobjective optimization problems. Asia Pac J Oper Res 2004;21:225–40.
[58] Abbass H, Sarker R, Newton C. PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on; 2001. p. 971–8.
[59] Coello CAC, Lechuga MS. MOPSO: a proposal for multiple objective particle swarm optimization. In: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on; 2002. p. 1051–6.
[60] Knowles JD, Corne DW. Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 2000;8:149–72.
[61] Liu D, Tan KC, Huang S, Goh CK, Ho WK. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. Eur J Oper Res 2008;190:357–82.
[62] Santana RA, Pontes MR, Bastos-Filho CJ. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. In: Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on; 2009. p. 237–42.
[63] Tripathi PK, Bandyopadhyay S, Pal SK. Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. Inf Sci 2007;177:5033–49.
[64] Raquel CR, Naval PC Jr. An effective use of crowding distance in multiobjective particle swarm optimization. In: Proceedings of the 7th Annual conference on Genetic and Evolutionary Computation; 2005. p. 257–64.
[65] Sierra MR, Coello CAC. Improving PSO-based multi-objective optimization using crowding, mutation andϵ-dominance. In: Evolutionary Multi-Criterion Optimization; 2005. p. 505–19.
[66] Mostaghim S, Teich J. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003. IEEE; 2003. p. 26–33.
[67] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. Evol Comput IEEE Trans 2002;6:182–97.
[68] Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: International Conference on Parallel Problem Solving From Nature. Springer; 2000. p. 849–58.
[69] Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 2015;89:228–49.
[70] Mirjalili S. The ant lion optimizer. Adv Eng Softw 2015;83:80–98.
[71] Mirjalili S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 2015:1–21.
[72] Mirjalili S, Mirjalili S, Hatamlou A. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 2015:1–19 /03/17 2015.
[73] Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 2016.
[74] Madin L. Aspects of jet propulsion in salps. Can J Zool 1990;68:765–77.
[75] Anderson PA, Bone Q. Communication between individuals in salp chains II. physiology. Proc R Soc Lond B 1980;210:559–74.
[76] Andersen V, Nival P. A model of the population dynamics of salps in coastal waters of the Ligurian Sea. J Plankton Res 1986;8:1091–110.
[77] Henschke N, Smith JA, Everett JD, Suthers IM. Population drivers of a Thalia democratica swarm: insights from population modelling. J Plankton Res 2015 p. fbv024.
[78] Coello CAC, Pulido GT, Lechuga MS. Handling multiple objectives with particle swarm optimization. Evol Comput IEEE Trans 2004;8:256–79.
[79] Digalakis J, Margaritis K. On benchmarking functions for genetic algorithms. Int J Comput Math 2001;77:481–506.
[80] M. Molga and C. Smutnicki, "Test functions for optimization needs," 2005.
[81] Yang X-S. Test problems in optimization; 2010. arXiv preprint arXiv:1008.0549.
[82] Kumar V, Chhabra JK, Kumar D. Effect of harmony search parameters' variation in clustering. Procedia Technol 2012;6:265–74.
[83] Kumar V, Chhabra JK, Kumar D. Automatic data clustering using parameter adaptive harmony search algorithm and its application to image segmentation. J Intell Syst 2016;25:595–610.
[84] Kumar V, Chhabra JK, Kumar D. Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems. J Comput Sci 2014;5:144–55.
[85] Kumar V, Chhabra JK, Kumar D. Variance-based harmony search algorithm for unimodal and multimodal optimization problems with application to clustering. Cybern Syst 2014;45:486–511.
[86] Kumar V, Chhabra JK, Kumar D. Clustering using modified harmony search algorithm. Int J Comput Intell Stud 2 2014;3:113–33.
[87] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff, "COCO: a platform for comparing continuous optimizers in a black-box setting," arXiv preprint arXiv:1603.08785, 2016.
[88] Hansen N, Auger A, Finck S, Ros R. Real-parameter black-box optimization benchmarking 2010: experimental setup. INRIA; 2010.
[89] Finck S, Hansen N, Ros R, Auger A. Real-parameter black-box optimization benchmarking 2010: presentation of the noisy functions. Research Center PPE; 2010. Technical Report 2009/21.
[90] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 2000;8:173–95.
[91] Zhang Q, Zhou A, Zhao S, Suganthan PN, Liu W, Tiwari S. Multiobjective optimization test instances for the CEC 2009 special session and competition. Multiobjective optimization test instances for the CEC 2009 special session and competition, 264. University of Essex, Colchester, UK and Nanyang technological University, Singapore; 2008. Special session on performance assessment of multi-objective optimization algorithms, technical report.
[92] Gandomi AH, Yang X-S, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 2013;29:17–35.
[93] Zhang M, Luo W, Wang X. Differential evolution with dynamic stochastic selection for constrained optimization. Inf Sci 2008;178:3043–74.
[94] Liu H, Cai Z, Wang Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 2010;10:629–40.
[95] Ray T, Saini P. Engineering design optimization using a swarm with an intelligent information sharing among individuals. Eng Optim 2001;33:735–48.
[96] Tsai J-F. Global optimization of nonlinear fractional programming problems in engineering design. Eng Optim 2005;37:399–409.
[97] Wang G-G, Guo L, Gandomi AH, Hao G-S, Wang H. Chaotic Krill Herd algorithm. Inf Sci 2014.
[98] Carlos A, COELLO C. Constraint-handling using an evolutionary multiobjective optimization technique. Civil Eng Syst 2000;17:319–46.
[99] Deb K. Optimal design of a welded beam via genetic algorithms. AIAA J 1991;29:2013–15.
[100] Deb K. An efficient constraint handling method for genetic algorithms. Comput Method Appl Mech Eng 2000;186:311–38.
[101] Krohling RA, dos Santos Coelho L. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. Syst Man Cybern Part B IEEE Trans 2006;36:1407–16.
[102] Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 2005;194:3902–33.
[103] Ragsdell K, Phillips D. Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind 1976;98:1021–5.
[104] He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 2007;20:89–99.
[105] Coello Coello CA. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Method Appl Mech Eng 2002;191:1245–87.
[106] Coello Coello CA, Mezura Montes E. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inf 2002;16:193–203.
[107] Siddall JN. Analytical decision-making in engineering design. Englewood Cliffs, NJ: Prentice-Hall; 1972.
[108] Wang GG. Adaptive response surface method using inherited latin hypercube design points. J Mech Des 2003;125:210–20.
[109] Cheng M-Y, Prayogo D. Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 2014;139:98–112.
[110] Chickermane H, Gea H. Structural optimization using a new local approximation method. Int J Numer Methods Eng 1996;39:829–46.
[111] Arora JS. Introduction to optimum design. Academic Press; 2004.
[112] Belegundu AD. Study of mathematical programming methods for structural optimization. Diss Abstr Int Part B 1983;43:1983.
[113] Yang XS. Nature-inspired metaheuristic algorithms. Luniver Press; 2011.
[114] Mezura-Montes E, Coello CAC. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 2008;37:443–73.
[115] Coello Coello CA. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 2000;41:113–27.
[116] Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization. Comput Struct 2012;112:283–94.

[117] Mahdavi M, Fesanghary M, Damangir E. An improved harmony search algorithm for solving optimization problems. Appl Math Comput 2007;188:1567–79.
[118] Li L, Huang Z, Liu F, Wu Q. A heuristic particle swarm optimizer for optimization of pin connected structures. Comput Struct 2007;85:340–9.
[119] Drela M. XFOIL: An analysis and design system for low Reynolds number airfoils. In: Low Reynolds number aerodynamics. Springer; 1989. p. 1–12.
[120] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. In: ACM SIGGRAPH computer graphics; 1986. p. 151–60.
[121] B.M. Pinkebtom, "The characteristics of; f 8; related airfoil sections from tests in the variable-density wind tunnel," 1933.
[122] Mirjalili S, Lewis A, Mirjalili SAM. Multi-objective optimisation of marine propellers. Procedia Comput Sci 2015;51:2247–56.
[123] Carlton J. Marine propellers and propulsion. Butterworth-Heinemann; 2012.
[124] Zeng Z-b, Kuiper G. Blade section design of marine propellers with maximum cavitation inception speed. J Hydrodyn Ser. B 2012;24:65–75.
[125] Epps B, Chalfant J, Kimball R, Techet A, Flood K, Chryssostomidis C. OpenProp: an open-source parametric design and analysis tool for propellers. In: Proceedings of the 2009 Grand Challenges in Modeling & Simulation Conference; 2009. p. 104–11.