Salp Swarm Optimization: a Critical Review

Mauro Castelli[1], Luca Manzoni[2], Luca Mariot[3], Marco S. Nobile[4,5,6], and Andrea Tangherloni[7]

[1] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312, Lisboa, Portugal ,
mcastelli@novaims.unl.pt

[2] Department of Mathematics and Geosciences, University of Trieste, 34127 Trieste, Italy ,
lmanzoni@units.it

[3] Cyber Security Research Group, Delft University of Technology, 2628 CD Delft, The Netherlands ,
l.mariot@tudelft.nl

[4] Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, 5612 AZ, Eindhoven, The Netherlands ,
m.s.nobile@tue.nl

[5] SYSBIO/ISBE.IT Centre of Systems Biology, 20126, Milan, Italy

[6] Bicocca Bioinformatics, Biostatistics and Bioimaging Centre (B4), 20854, Vedano al Lambro, Italy

[7] Department of Human and Social Sciences, University of Bergamo, 24129, Bergamo, Italy ,
andrea.tangherloni@unibg.it

May 7, 2022

Abstract

In the crowded environment of bio-inspired population-based metaheuristics, the Salp Swarm Optimization (SSO) algorithm recently appeared and immediately gained a lot of momentum. Inspired by the peculiar spatial arrangement of salp colonies, which are displaced in long chains following a leader, this algorithm seems to provide an interesting optimization performance. However, the original work was characterized by some conceptual and mathematical flaws, which influenced all ensuing papers on the subject. In this manuscript, we perform a critical review of SSO, highlighting all the issues present in the literature and their negative effects on the optimization process carried out by this algorithm. We also propose a mathematically correct version of SSO, named Amended Salp Swarm Optimizer (ASSO) that fixes all the discussed problems. We benchmarked the performance of ASSO on a set of tailored experiments, showing that it is able to achieve better results than the original SSO. Finally, we performed an extensive study aimed at understanding whether SSO and its variants provide advantages compared to other metaheuristics. The experimental results, where SSO cannot outperform simple well-known metaheuristics, suggest that the scientific community can safely abandon SSO.

1

Keywords Metaheuristics; Global Optimization; Bound Constrained Optimization; Shift Invariant Functions.

1 Introduction

Most of the real-world problems can be formulated in terms of optimization problems, that is, they can be solved by finding the values that lead to the maximization, or minimization, of a target fitness function. However, an analytic solution for such function is seldom available; moreover, these problems are often high-dimensional and are characterized by several local optima, where simple optimization algorithms easily get stuck. In this context, bio-inspired population-based global optimization metaheuristics often proved their effectiveness in the identification of optimal solutions (i.e., global optima).

A very large number of metaheuristics have been proposed in the latter years, taking inspiration among others from: Darwinian processes such as Differential Evolution [53], Evolution Strategies [31], Genetic Algorithms [32], and all related variants [66]; the emergent collective behavior of groups of animals, including Particle Swarm Optimization [20, 52], Artificial Bee Colony [37], Ant Colony Optimization [18], Bat Swarm Optimization [74], Firefly Algorithm [76], and all related variants [46]); finally, natural dynamic phenomena such as Gravitational Search [55] and Artificial Immune Systems [12].

Recently, a metaheuristics named Salp Swarm Optimization (SSO), which is inspired by salp colonies, was proposed in [43]. Salps are barrel-shaped animals living in swarms that are organized in long chains, actively looking for phytoplankton. SSO leverages the metaphor of this peculiar spatial arrangement by translating it into an optimization algorithm. Specifically, the individuals composing a population are constrained to follow the leader salp, while the latter is performing the actual exploration for food sources. In this metaphor, the leader explores the search space for optimal regions—with respect to the fitness function—while the followers exploit the area surrounding the leader.

SSO quickly became popular as a real-valued [2], discrete [6, 23], and even multi-objective optimization algorithm [36]. This popularity apparently stems both from its performance and peculiar functioning paradigm, making the method to stand out among a plethora of seemingly novel algorithms [17].

In this paper, we show that the original algorithm is characterized by some conceptual and implementation flaws, hampering the performance of SSO and all derived algorithms under some specific circumstances. In particular, we will show that the most remarkable issue is the fact that SSO, in its original formulation, is not shift-invariant. This raises a problem if the lower bound of the search space is too far from the origin. In this paper, we first propose an Amended Salp Swarm Optimizer (ASSO), which represents a correct reformulation of the SSO algorithm as was probably intended by its creators. Still, our objective is not to promote the use of ASSO in the scientific community. Indeed, while ASSO fixes the mathematical flaws of SSO, it is still not grounded on any theoretical basis. Thus, ASSO would be one of the several metaheuristics based on some natural metaphor and built by combining search operators that do not rely on any theoretical property [49]. For this reason, we subsequently assess the performance of ASSO and SSO against simple and well-known metaheuristics on a set of benchmark functions. This analysis aims at understanding whether the scientific community could abandon the use of SSO and ASSO, or if these algorithms provide some advantages with respect to simpler metaheuristics widely established in this research field. As a side

note, SSO changed name several times in the literature: indeed, we can find it as Salp Swarm Algorithm [56], Salp Swarm Optimizer [22, 58], Salp Swarm Optimization [11], Salp Optimization Algorithm [64], and so forth. Incidentally, in the highlights of the original paper [43], the authors state that the official name of the algorithm is Salp Swarm Optimizer, although the acronym SSA is used in the rest of the article. In this paper, we will use the name Salp Swarm Optimization and the acronym SSO to prevent confusion. We wish to emphasize that this work aims at discussing the limitations and issues of the original SSO algorithm, showing how even a simple Random Search (RS) algorithm can outperform the original incorrect SSO algorithm, under specific circumstances. Therefore, we think that our results and findings raise an alert on the existing SSO literature.

This paper is structured as follows.

Section 2 reviews the related works highlighting the flaws and limitations of some of the recently defined metaheuristics. Section 3 provides a critical review of SSO, in particular concerning the methodological issues with the updating rule of the position of the leader salp and the physically-inspired motivation for the updating rule of the positions of the follower salps. Section 4 presents an experimental evaluation of the improved version of SSO that fixes some of the raised issues, and compares the result with the original version on a set of standard benchmark functions. Subsequently, we analyze the performance of SSO and its variants on the CEC 2017 benchmark function suite against two commonly used metaheuristics, namely Covariance Matrix Adaptation Evolution Strategy (CMAES) [30] and Differential Evolution (DE) [60]. Finally, Section 5 summarises the main issues of the original SSO algorithm, and concludes with a broader remark on the nature of the salp metaphor.

2   Related work

Recent years have witnessed the definition of a significant number of metaheuristics inspired by some natural phenomenon [24]. The common idea behind the definition of these metaheuristics is to consider a natural process and, subsequently, to design the underlying metaheuristic by exploiting the natural metaphor observed. After the publication of a given metaheuristic, it is also common to see a significant number of scientific papers that use the new metaheuristic to address complex real-world problems claiming its superior performance compared to existing metaheuristics. Luckily, a research strand started to criticize the definition of these metaheuristics, observing that, in most of the cases, their performance cannot be better than the commonly used evolutionary strategies [69]. Despite the lack of novelty that characterizes these metaheuristics (i.e., the change concerns only the underlying natural metaphor), a fundamental issue is that some of the results achieved through them are not reliable.

In this sense, one of the clearest examples is the paper by Weyland [69], in which the author demonstrated that Harmony Search (HS) [39] cannot be used to successfully solve a sudoku, thus contradicting the results obtained by Geem [27]. More precisely, Weyland first proved that HS is a special case of evolution strategies, thus highlighting the lack of novelty of the metaheuristic. As a consequence, the performance of HS is always bounded by the performance that can be obtained by evolution strategies. Finally, Weyland demonstrated that the results achieved in [27] are flawed both from the theoretical and the practical point of view, concluding that there is no reason for the existence of HS as a novel metaheuristic. Despite the Weyland's work clearly demonstrated the lack of novelty of HS, the algorithm is still largely used nowadays. Further, even though

Weyland proved beyond any doubt that HS cannot perform better than evolutionary strategies, several papers still claim its presumed superior performance [5].

Thus, it seems that practitioners are still deceived by metaheuristics whose novelty is based only on the use of some natural metaphors. The truth is that HS, and other related metaheuristics, are simply using a different terminology with respect to the classic evolutionary strategies. While the lack of novelty did not prevent metaheuristics based on natural metaphors to be published in well-renowned scientific venues, some scientific journals are seriously tackling the problem. For instance, Marco Dorigo, the editor in chief of Swarm Intelligence, published an editorial note [16] stating that he observed a new trend consisting in "taking a natural system/process and use it as a metaphor to generate an algorithm whose components have names taken from the natural system/process used as metaphor". Dorigo also highlighted that "this approach has become so common that there are now hundreds of so-called new algorithms that are submitted (and unfortunately often also published) to journals and conferences every year", and concluded his editorial stating that "it is difficult to understand what is new and what is the same as the old with just a new name, and whether the proposed algorithm is just a small incremental improvement of a known algorithm or a radically new idea".

A similar analysis on this trend appeared in the work by Cruz *et al.* [14]. There, the authors first highlighted the vast number of swarm intelligence algorithms developed by taking inspiration from the behavior of insects and other animals and phenomena. Subsequently, they showed that most algorithms present common macro-processes among them, despite the fact that they are inspired by different metaphors. In other words, the considered metaheuristics are characterized by common issues and features which happen at the individual level, promoting very similar emergent phenomena [14]. Thus, it is difficult (if not impossible) to claim that such metaheuristics are really novel.

Focusing on specific metaheuristics, some contributions that analyze the behavior of a given algorithm started to appear [45, 67, 68]. In [68], Villalón *et al.* thoroughly investigated the Intelligent Water Drops (IWD) algorithm [33], a metaheuristic proposed to address discrete optimization problems. The authors demonstrated that the main steps of the IWD algorithm are special cases of Ant Colony Optimization (ACO) [18]. Thus, the performance of IWD cannot be better than the best ACO algorithm. Moreover, the authors analyzed the metaphor used for the IWD definition, and from their analysis, the metaphor is based on "unconvincing assumptions of river dynamics and soil erosion that lack a real scientific rationale". Finally, they pointed out that the improvements proposed to the IWD algorithm are based on ideas and concepts already investigated in the literature many years before in the context of ACO.

Niu *et al.* [45] analyzed the Grey Wolf Optimization (GWO) algorithm [44], and demonstrated that, despite its popularity, GWO is flawed. In particular, GWO shows good performance for optimization problems whose optimal solution is 0, while the same performance cannot be obtained if the optimal solution is shifted. In particular, when GWO solves the same optimization function, the farther the function's optimal solution is from 0, the worse its performance is. Interestingly, GWO was proposed by the same author of the SSO algorithm analyzed in this paper, and it presents some of the SSO's flaws.

Villalón *et al.* [67] analyzed the popular Cuckoo Search (CS) algorithm [75], a metaheuristic introduced in 2009. The authors analyzed CS from a theoretical standpoint, and showed that it is based on the same concepts as those proposed in the $(\mu + \lambda)$ evolution strategy proposed in 1981 [57]. Further, the authors evaluated the algorithm and the metaphor used for its definition based on four criteria (i.e., usefulness, novelty,

dispensability, and sound motivation), and they concluded that CS does not comply with any of these criteria. Finally, they pointed out that the original CS algorithm does not match the publicly available implementation of the algorithm provided by the authors of the algorithm.

This analysis is quite surprising, given the popularity of CS, and it highlights the need for a thorough investigation of the existing metaheuristics, with the goal of understanding which of them should be abandoned by the scientific community. Indeed, the impressive number of metaheuristics published in the literature makes it difficult to determine whether they really contribute to the advancement of the field. This problem was pointed out by Martínez *et al.* [26]. Specifically, the authors investigated whether the increasing number of publications is correlated with real progress in the field of heuristic-based optimization. To answer this research question, the authors compared five heuristics proposed in some of the most reputed journals in the area, and compared their performance to the winner of the IEEE Congress on Evolutionary Computation 2005. The results showed that the considered methods could not achieve the result of the competition winner, which was published several years before. Moreover, a comparison with the state-of-the-art algorithms is often missing, thus making it impossible to understand the real advantage provided by a new method.

In the same vein, Piotrowski and Napiorkowski [49] highlighted the risk associated with the definition of new and increasingly complex optimization metaheuristics and the introduction of structural bias [38]. In particular, the authors focused on two winners of the CEC 2016 competition, and they found out that each of them includes a procedure that introduces a structural bias by attracting the population towards the origin. As a final message, the authors highlighted that some metaheuristics have to be simplified because they contain operators that structurally bias their search, while other metaheuristics should be simplified (or abandoned) as they use unnecessary operators.

3 Critical review of SSO

This section outlines the design issues we found in the original definition and implementation of the SSO algorithm. More precisely, the following issues are discussed here and in Section 4:

- the update rule of the main salp does not work correctly when one of the dimensions has a lower bound different from zero (Section 3.1). In the experimental part, we show that simply shifting a 2D sphere function from the origin makes SSO perform worse than a simple RS (Section 4);

- the physical motivations for the updating rule of the follower salps are incorrectly derived from Newton's laws of motion (Section 3.2);

- there is a clear divergence between the algorithm described in [43] and the available SSO implementation. This makes it difficult, or even impossible, to compare results from different papers (Section 3.3);

- finally, we experimentally show that the original SSO algorithm has a bias towards the origin. Since many of the considered benchmark problems have the optimum in the origin, the results are biased in favor of SSO (Section 4).

In what follows, we will refer to the equations that were introduced in the original SSO paper by [43].

3.1 Leader Salp Updating Rule

We assume here that a chain of N different salps moves within a bounded D-dimensional search space, aiming at identifying the optimal solution.

The first issue is related to the definition of the updating rule for the position of the leader salp. In Equation 3.1 of the original paper [43], the update for the leader salp along the j-th dimension, with $j = 1, \ldots, D$, is given as follows:

$$x_j^1 = \begin{cases} F_j + c_1((ul_j - lb_j)c_2 + lb_j) & \text{if } c_3 \geq 0 \\ F_j - c_1((ul_j - lb_j)c_2 + lb_j) & \text{if } c_3 < 0 \end{cases} , \tag{1}$$

where the upper and lower bounds of the search space in the j-th dimension are ul_j and lb_j, respectively. The value $F_j \in [lb_j, ul_j]$ is the position of the best solution found so far in the j-th dimension, that corresponds to the best food source. On the other hand, c_1 decreases exponentially with the number of iterations according to the following rule:

$$c_1 = 2e^{-(\frac{4\ell}{L})^2} ,$$

where ℓ is the current iteration number while L is the total number of iterations. Finally, both c_2 and c_3 are random numbers selected uniformly in the range $[0, 1]$.

Here, we can find the first definitional issue, although it did not propagate in the original source code. Indeed, $c_3 \in [0, 1]$ implies that the second case of Equation (1) is never verified. However, in the source code the threshold for c_3 is set to 0.5 instead of 0, which means that the two cases of the updating rule occur with equal probability. This kind of oversights is usually not a problem, but several successive papers *do not correct* the issue (see, e.g., [4, 8, 21, 34, 35, 56, 65, 72, 73]). Further, if a new implementation following the specifications of [43] is used, instead of the original one released by the authors, the results might not be comparable. It is currently unknown how many papers on SSO employ the same implementation of the original paper.

The main issue with the updating rule is however more significant, and it is still present in the source code and widespread across all existing SSO variants ([1, 3, 4, 7, 8, 21, 22, 34, 35, 54, 56, 65, 72, 73]). Let us consider the variation with respect to F_j, which is:

$$c_1((ul_j - lb_j)c_2 + lb_j) = c_1 c_2(ul_j - lb_j) + c_1 lb_j .$$

Notice that while $c_1 c_2(ul_j - lb_j)$ gives a value in $[0, ul_j - lb_j]$ (i.e., $F_j + c_1 c_2(ul_j - lb_j)$ might remain inside the search space), the value $c_1 lb_j$ is also added and it can be arbitrarily large in absolute value. Although this is not the case in the experiments performed in the original paper, simply shifting the search space by a suitably large constant might significantly hamper the search process of SSO. The issue could also potentially affect research in applied disciplines where SSO has been used (e.g., COVID-19 related research, as proposed by [4]).

As an example, let us suppose that the lower and the upper bounds are of the same order of magnitude for all the D dimensions, and in particular that $lb_j = 10^k$ and $ub_j = 10^k + 1$, with $j = 1, \ldots, D$. In other words, the search space is the hypercube $[10^k, 10^k + 1]^j$. By using the updating rule given in Equation (1), the position of the leader salp is updated as

$$\begin{aligned} x_j^1 &= F_j \pm c_1 c_2(ul_j - lb_j) + c_1 lb_j \\ &= F_j \pm c_1 c_2 + c_1 10^k \end{aligned} . \tag{2}$$

Since $c_1 c_2 \leq 2$ (recall that $c_2 \in [0, 1]$), the position update in Equation (2) is dominated by the $c_1 10^k$ term. Consequently, this update will move the leader salp out of the admissible bounds for most of the values taken by c_1, forcing its position to be clipped on the borders of the search space most of the times. This implies that SSO is not invariant with respect to translations of the search space. Indeed, given that $c_1 = 2e^{-(\frac{4\ell}{L})^2}$, the salp remains inside the search space $[10^k, 10^k + 1]^j$ only if $c_1 10^k \leq 1$, namely:

$$2e^{-(\frac{4\ell}{L})^2} 10^k \leq 1 \implies k \leq \log_{10}\left(\frac{1}{2} e^{-(\frac{4\ell}{L})^2}\right) .$$

The effect is that, depending on the search space, the majority of the updates of the leader salp can force it on the boundary of the space (due to clipping), with only the later iterations (with c_1 small enough) resulting in the leaders salp moving without clipping. In particular $e^{-(\frac{4\ell}{L})^2}$ yields the smallest value when $\ell = L$, i.e., at the last iteration and in this case we obtain that

$$k \leq \log_{10}\left(\frac{1}{2} e^{-16}\right) \approx 6.648 .$$

Therefore, $k > 6.648$ will result in a search space where the position of the leader salp will *always* be forced outside of the search space, or equivalently, the leader salp will continue to "bounce" on the boundaries of the search space.

Similar pathological examples can be found by tweaking the values of the upper and lower bounds. This observation reveals that the initial value of c_1 must be carefully chosen with respect to the size and the shift of the search space. In other words, SSO is also not invariant with respect to rescalings of the search space.

Another different issue is that for any dimension j, the quantity $c_2(ul_j - lb_j) + lb_j$ has an expected value of $\frac{ul_j + lb_j}{2}$. When the search space is centered in zero the expected value is then zero. As we will see in the experimental part, this gives an unfair advantage to problems where the search space is symmetric (with respect to $\mathbf{0}$) and the global optimum is in $\mathbf{0}$.

3.2 Physically-inspired motivations

The original paper claimed that the definition of the updating rule for the follower salps is based on the principles of classical mechanics (Newton's laws). However, there are important issues concerning the formulation of this rule, as well as the correct use of Newton's laws of motion. The equation for the update of the follower salps in SSO is:

$$x_j^i = \frac{1}{2} a t^2 + v_0 t , \tag{3}$$

where it can be assumed that $a = a_j^i$ and $v_0 = v_j^i$ at $t = 0$, for each dimension j (with $j = 1, \ldots, D$) and follower salp i (with $i = 1, \ldots, N$). In the original paper, the acceleration a is calculated as:

$$a = \frac{v_{\text{final}}}{v_0} , \tag{4}$$

which is incorrect, since the average acceleration in a time interval Δt is:

$$a = \frac{v(t + \Delta t) - v(t)}{\Delta t} .$$

Notice that the incorrect formula also gives a dimensionless quantity, instead of a length divided by a squared time. Notably, this issue is only sometimes corrected (with $a = v_{\text{final}} - v_0$) in the SSO literature.

In addition, the formula $v_{final} = \frac{x - x_0}{t}$, with x and x_0 being the final and initial positions, respectively, and t being the time interval, is correct for the *average speed*, which is however not necessary in all the derivations of the paper. In fact, for computing the average acceleration, the *instantaneous speed* must be used instead. Furthermore, the aim of the derivation is to obtain the final position of the salp, which we cannot use to compute the average speed. Since the salps are initially still, the original definition of SSO [43] explicitly uses $v_0 = 0$ which, when substituted in the Equations 3 and 4, gives an infinite acceleration that would lead the salps outside any boundary of the search space.

Regardless of these issues, the authors eventually point out that t, in the equations, corresponds to the iteration number, so that $\Delta t = 1$ and the time term can be cancelled out from the equations. They conclude that this step leads to the final formula:

$$x^i_j = \frac{1}{2}(x^i_j + x^{i-1}_j) \ .$$ (5)

Unfortunately, this equation cannot be derived from the previous ones. In fact, the position of the next salp in the chain never appears before this point and it is not taken into account in any of the previous derivations.

A correct way to derive the previous formula would be the following. Assume that the i-th salp is moving toward the current position of the $(i-1)$-th salp (in the j-th dimension) with starting speed of 0, final speed of 1 unit for time step, and constant acceleration of $(x^i_j(t) - x^{i-1}_j(t))$ units for time step squared. Hence, the new position after one time unit can be computed as follows:

$$\begin{aligned} x^i_j(t+1) &= \frac{1}{2}(x^{i-1}_j(t) - x^i_j(t)) + x^i_j(t) \\ &= \frac{1}{2}(x^i_j(t) + x^{i-1}_j(t)). \end{aligned}$$

Notice that this is only one of the possible ways to *correctly* derive the updating rule for the follower salps, but it is completely unrelated to the biological metaphor exploited by the authors. This critique only concerns the physical motivations for the definition of the updating rules, not the updating rule itself. However, the flawed explanation presented in the original paper is restated into multiple papers [34, 35, 65, 72], without any significant correction.

3.3 Implementation Issues

It is worth mentioning that both the MATLAB® and Python implementations[1] do not correctly implement the pseudo-code and, unfortunately, do not follow the explanations provided by the authors in the original paper [43]. As a matter of fact, in both implementations, the authors updated the first half of the population using Equation 1 and the second half using Equation 5. Considering that the salp chain is composed of N different individuals, the first $\frac{N}{2} - 1$ salps perform the exploration process for food sources attracted by the best food source found so far (updated by Equation 1). The

---

[1]The source code of both implementations is available at the following address: `http://www.alimirjalili.com/SSA.html`

$\frac{N}{2}$-th individual is the leader salp that drags the follower salps, which exploit the area surrounding the leader (updated by Equation 5). In what follows, we will refer to this implementation as SSO-code.

We modified the implementation of SSO-code by removing the term $c_1 lb_j$ from Equation (1), for each dimension j (with $j = 1, \ldots, D$). In such a way, we are able to avoid as much as possible the clipping step of the salp positions due to the wrong update, proposed in the original Equation (i.e., Equation 1), which sends the salps out of the admissible bounds. We will refer to this version as ASSO.

4   Experimental Evaluation

As a first batch of tests, we compared the performance of a simple Random Search (RS), SSO, SSO-code, and ASSO using 2-*D* standard benchmark functions (i.e., Ackley, Alpine, Rosenbrock, and Sphere). Then, the RS, SSO, SSO-code, and ASSO were compared against basic versions of CMAES [30] and DE [60]. We used the standard version of CMAES and DE implemented in the Pymoo library (Multi-objective Optimization in Python) [10], which allows for easily using both single- and multi-objective algorithms, exploiting the default parameters proposed by the authors of the library. Specifically, concerning CMAES, the initial standard deviation was set to 0.5 for each coordinate, and no restart strategy was applied. Regarding DE, according to the classic DE taxonomy, the DE/rand/1/bin variant was used with a differential weight $F = 0.3$ and a crossover rate equal to 0.5.

4.1   Standard benchmark functions

To show that both SSO and SSO-code are not shift-invariant, the search spaces of the tested benchmark functions were shifted by a large constant (i.e., 10^9). To collect statistically sound results, for each function, we ran the tested techniques 30 times. For each iteration, we kept track of the fitness value of the best individual, over the 30 repetitions, to calculate the *Average Best Fitness* (ABF). For a completely fair comparison among the different techniques, we fixed a budget of 100 iterations using 50 individuals. Note that the implemented RS randomly generates 50 particles, at each iteration, without taking into account any information of the previous iterations.

Figure 1a clearly shows that both SSO and SSO-code are not shift-invariant. Indeed, shifting 2-*D* standard benchmark functions by a large constant hampered the optimization abilities of both SSO and SSO-code. Across all the tested functions, even RS obtained better results compared to SSO and SSO-code. On the contrary, our proposed algorithm ASSO – where we simply removed the term $c_1 lb_j$ – was able to outperform the other techniques. It is worth reminding that ASSO is not a novel algorithm, but an amended version of SSO in which the mathematical errors have been corrected.

In order to evaluate whether the achieved results were different also from a statistical point of view, we applied the Mann–Whitney U test with the Bonferroni correction [19, 42, 70]. Specifically, we applied this statistical test to independently compare the results obtained by the techniques on each benchmark function. Thus, for each benchmark function and for each technique, we built a distribution by considering the fitness value of the best individual at the end of the last iteration over the 30 repetitions. The boxplots in Figure 1b show the distribution of the best fitness values, achieved at the end of the executions of the tested techniques. Figure 1b also reports the results of the statistical tests by using the asterisk convention. These results indicate that,
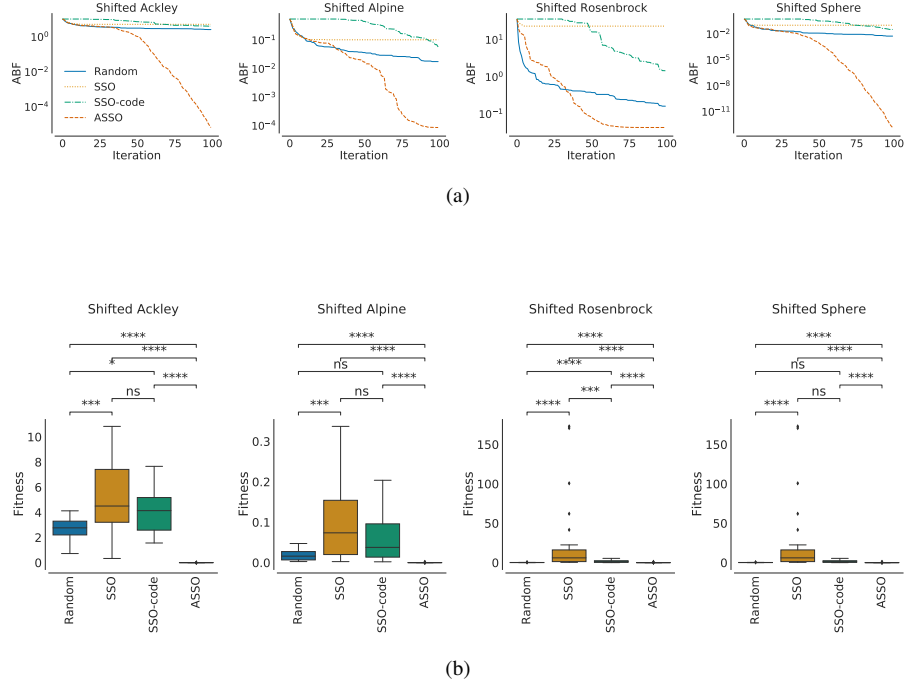
(a)

(b)

Figure 1: **(a)** Comparison of the ABF obtained by the analyzed techniques for each tested function. The ABF was calculated by using the fitness values of the best individual over the 30 repetitions. To better appreciate the differences among the tested techniques, the *y*-axes are on a logarithmic scale. **(b)** The boxplots show the distribution of the last fitness value of the best individual over the 30 repetitions. p-value≤ 0.0001 (****); $0.0001 <$ p-value≤ 0.001 (***); $0.001 <$ p-value≤ 0.01 (**); $0.01 <$ p-value≤ 0.05 (*); p-value> 0.05 (ns).

generally, there is no statistical difference among SSO and SSO-code. Only considering the Rosenbrock function a strong statistical difference is present between SSO and SSO-code. The simple RS obtained similar or even better results than SSO and SSO-code, while there is always a strong statistical difference between the results achieved by ASSO and those achieved by the other techniques, demonstrating the effectiveness of our correction.

Concerning the problem described in Section 3.1, we show that, on a search space symmetric with respect to zero, the SSO algorithm has a bias toward the origin. In order to show this behavior, we will use a fitness function that returns a random number with uniform distribution in $[0, 1)$. For a swarm intelligence algorithm, we would expect a uniform distribution of particles across the entire search space using such a fitness function. Stated otherwise, using a random fitness function, the salps should not converge anywhere, and should randomly wander across the search space. However, we will show that, following the non-amended equations provided in the original SSO paper [43], the swarm converges in the origin providing an unfair advantage in the case of optimization problems whose global optimum lies in $\mathbf{x} = \mathbf{0}$ (and possibly lead to sub-optimal performance in the case of real-world functions).

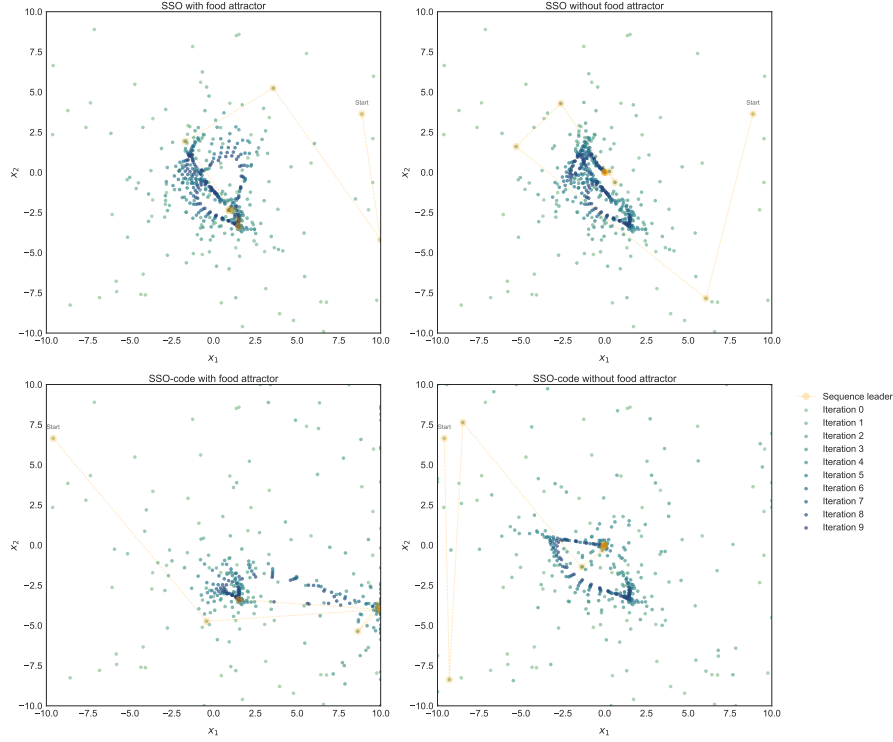Figure 2 shows the result of this test, performed on both SSO and SSO-code with

Figure 2: Dynamic behavior of the salp swarm: the leader leaves the feasible region, is relocated on the boundary, and then the swarm is attracted towards the origin of axis.

and without the food attractor. The Figure reports the positions of all salps during the first 10 iterations of the optimization; the position of the leader salp is highlighted by an orange circle, and the initial position is denoted by the text "Start". According to our results, the leader salp get attracted toward zero. The same happens for SSO-code: the leader salp it is inevitably attracted towards the center of the search space. The attraction towards the origin is even more evident in the case of SSO and SSO-code without food attraction: the swarm perfectly converges to the origin and no longer moves.

4.2 CEC 2017 benchmark function suite

During the last years, different benchmark function suites have been proposed to test and compare existing and novel global optimization techniques [63]. The benchmark functions they contain try to mimic the behavior of real-world problems, which often show complex features that basic optimization algorithms might not be able to grasp [25, 63]. Regarding the real-parameter numerical optimization, the IEEE Congress on Evolutionary Computation (CEC) and the Genetic and Evolutionary Computation Conference (GECCO) include competitions where complex benchmark function suites have been designed to test and compare global optimization techniques [13, 15, 28, 29].

Here, we evaluated the performance of the considered optimization techniques using the CEC 17 benchmark problems for single-objective real-parameter numerical optimization [9], which were previously used to compare the performance of different

11

Table 1: Definitions and optimum values of the CEC 2017 benchmark functions designed for the competition on real-parameter single objective numerical optimization.

Typology	No.	Function name	Opt.
Unimodal Functions	1	Shifted and Rotated Bent Cigar	100
	2	Shifted and Rotated Sum of Different Power	200
	3	Shifted and Rotated Zakharov	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock	400
	5	Shifted and Rotated Rastrigin	500
	6	Shifted and Rotated Expanded Schaffer F6	600
	7	Shifted and Rotated Lunacek Bi-Rastrigin	700
	8	Shifted and Rotated Non-Continuous Rastrigin	800
	9	Shifted and Rotated Levy	900
	10	Shifted and Rotated Schwefel	1000
Hybrid Functions	11	Zakharov; Rosenbrock; Rastrigin	1100
	12	High-conditioned Elliptic; Modified Schwefel; Bent Cigar	1200
	13	Bent Cigar; Rosenbrock; Lunacek bi-Rastrigin	1300
	14	High-conditioned Elliptic; Ackley; Schaffer F7; Rastrigin	1400
	15	Bent Cigar; HGBat; Rastrigin; Rosenbrock	1500
	16	Expanded Schaffer F6; HGBat; Rosenbrock; Modified Schwefel	1600
	17	Katsuura; Ackley; Expanded Griewank plus Rosenbrock; Schwefel; Rastrigin	1700
	18	High-conditioned Elliptic; Ackley; Rastrigin; HGBat; Discus	1800
	19	Bent Cigar; Rastrigin; Griewank plus Rosenbrock; Weierstrass; Expanded Schaffer F6	1900
	20	HappyCat; Katsuura; Ackley; Rastrigin; Modified Schwefel; Schaffer F7	2000
Composition Functions	21	Rosenbrock; High-conditioned Elliptic; Rastrigin	2100
	22	Rastrigin; Griewank; Modified Schwefel	2200
	23	Rosenbrock; Ackley; Modified Schwefel; Rastrigin	2300
	24	Ackley; High-conditioned Elliptic; Griewank; Rastrigin	2400
	25	Rastrigin; HappyCat; Ackley; Discus; Rosenbrock	2500
	26	Expanded Schaffer F6; Modified Schwefel; Griewank; Rosenbrock; Rastrigin	2600
	27	HGBat; Rastrigin; Modified Schwefel; Bent Cigar; High-conditioned Elliptic; Expanded Schaffer F6	2700
	28	Ackley; Griewank; Discus; Rosenbrock; HappyCat; Expanded Schaffer F6	2800
	29	f_{15}; f_{16}; f_{17}	2900
	30	f_{15}; f_{18}; f_{19}	3000

metaheuristics [47, 62, 63]. Table 1 reports the tested benchmark problems, which are based on shifted, rotated, non-separable, highly ill-conditioned, and complex optimization benchmark functions [9]. We optimized each function f_k (with $k = 1, \ldots, 30$) considering the dimensions $D = \{10, 30\}$, search space boundaries $[-100, 100]^D$, and a maximum number of function evaluations $\texttt{MaxFES} = D \cdot 10^4$. For each technique, we executed 30 independent runs to collect statistically sound results.

Figure 3 depicts the Average Best Fitness (ABF) obtained by the analyzed techniques for each function f_k (with $k = 1, \ldots, 30$) and $D = 10$, showing that DE was able to obtain better results than all the SSO-based strategies, including ASSO, and the simple RS tested here. As one can see, DE outperformed the SSO-based strategies in 27 out of 30 functions. CMAES also obtained better results than the SSO-based strategies in more than half of the tested functions. As we did for the standard benchmark functions, we evaluated if the achieved results were also different from a statistical point of view by using the Mann–Whitney U test with the Bonferroni correction [19, 42, 70]. Thus, we independently compared the results obtained by the six techniques on each benchmark function f_k (with $k = 1, \ldots, 30$) considering the distributions built using the value of the best individual at the end of the last iteration over the 30 repetitions. The heatmaps showed in Figure 4 clearly point out that there is almost always a strong statistical difference between the results achieved by all the SSO-based strategies, including ASSO, and those obtained by DE. Specifically, SSO-based strategies were able to obtain comparable or better results than those reached by DE only for the functions f_{21}, f_{25}, and f_{28}. Comparing the results achieved by CMAES and those obtained by SSO-based
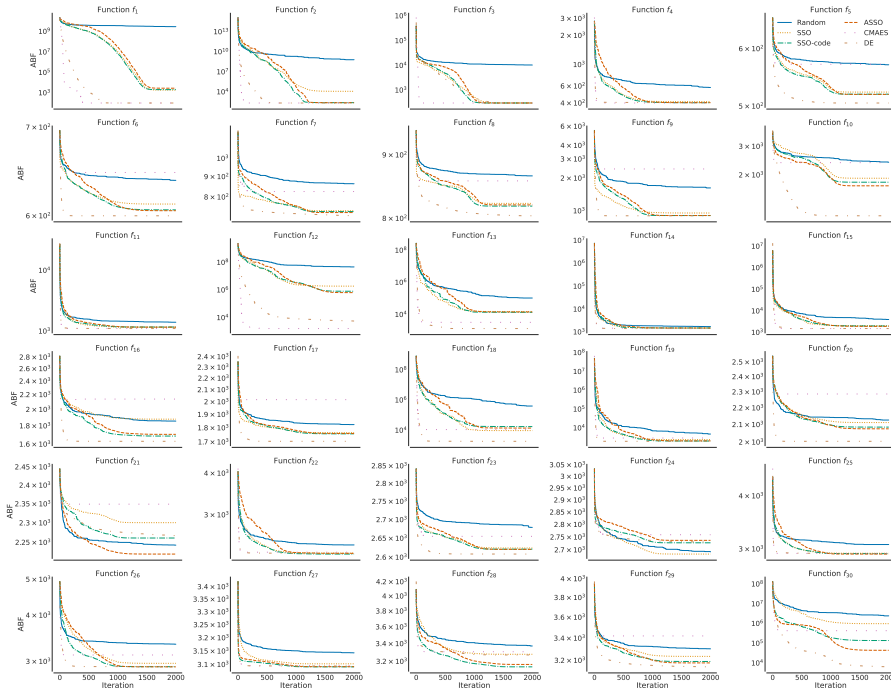
Figure 3: Comparison of the Average Best Fitness (ABF) obtained by the considered techniques for each function f_k (with $k = 1, \dots, 30$) and $D = 10$. The ABF was calculated by using the fitness values of the best individual over the 30 repetitions. Note that the y-axes are on a logarithmic scale.

strategies, there is a strong statistical difference in more than half of the tested functions. These results are coherent with the probabilistic re-formulation of the *no-free lunch theorem* (NFL) [40], an extension of the original version of the NFL theorem [41, 71], which prove the validity of the theorem in continuous domains. Thus, according to the NFL theorem, no algorithm outperforms all the competitors in any optimization problem.

Increasing the dimensions of the tested benchmark functions from 10 to 30 does not allow the SSO-based approaches to obtain better results than DE. As a matter of fact, Figures 5 and 6 clearly show that DE outperformed all the SSO-based approaches in 22 out of 30 benchmarks functions. In the remaining functions (i.e., f_4, f_7, f_{14}, f_{18}, f_{25}, f_{26}, f_{27}, and f_{28}) some SSO-based approaches (generally ASSO) obtained comparable results than those achieved by DE. However, there is no benchmark function where all the SSO-based methods were able to obtain similar results to those reached by DE, confirming the poor performance of all the tested variants of SSO.

5 Conclusion

This paper shows systematic and deep issues in the definition of a widely-cited optimization algorithm, namely Salp Swarm Optimization. In particular, the multiple issues concerning the updating rules, the physically-inspired motivations, and the in-
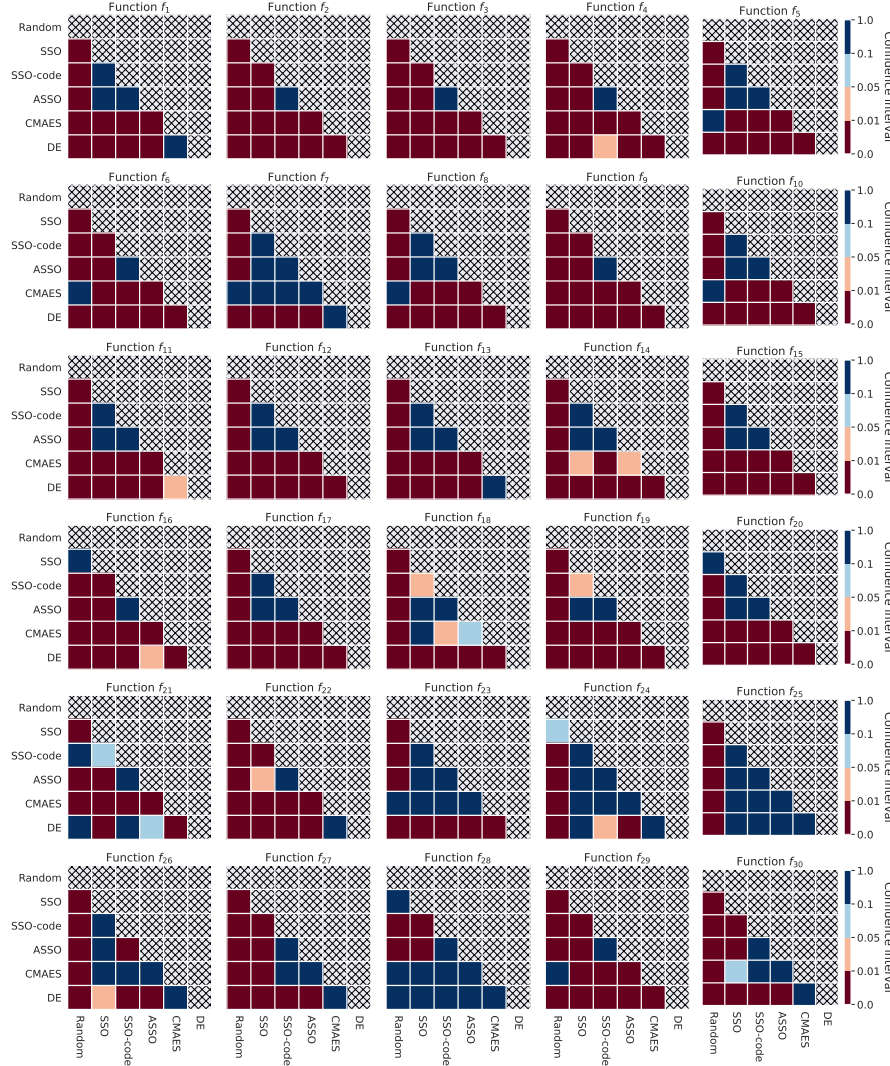
Figure 4: Heatmaps showing the p-values of the Mann–Whitney U test with the Bonferroni correction for each function f_k (with $k = 1, \ldots, 30$) and and $D = 10$. The confidence interval was divided into 4 levels indicating a strong statistical difference, statistical difference, weak statistical difference, and no statistical difference, respectively.

consistency between the code and the description in the original paper that proposed Salp Swarm Optimization raise concerns about all ensuing related literature, since the erroneous derivations and rules are present in most of the published papers on the topic. Furthermore, it is currently problematic to discern which results can be trusted, which ones are based on an incorrect implementation, and which papers have an incorrect description but are using a correct implementation.

The most serious issue analyzed in this paper is perhaps the presence of the lower bound lb_j in the updating rule of the leader salp (see Equation 1). In particular, this term makes the algorithm not shift-invariant, introducing a severe search bias that
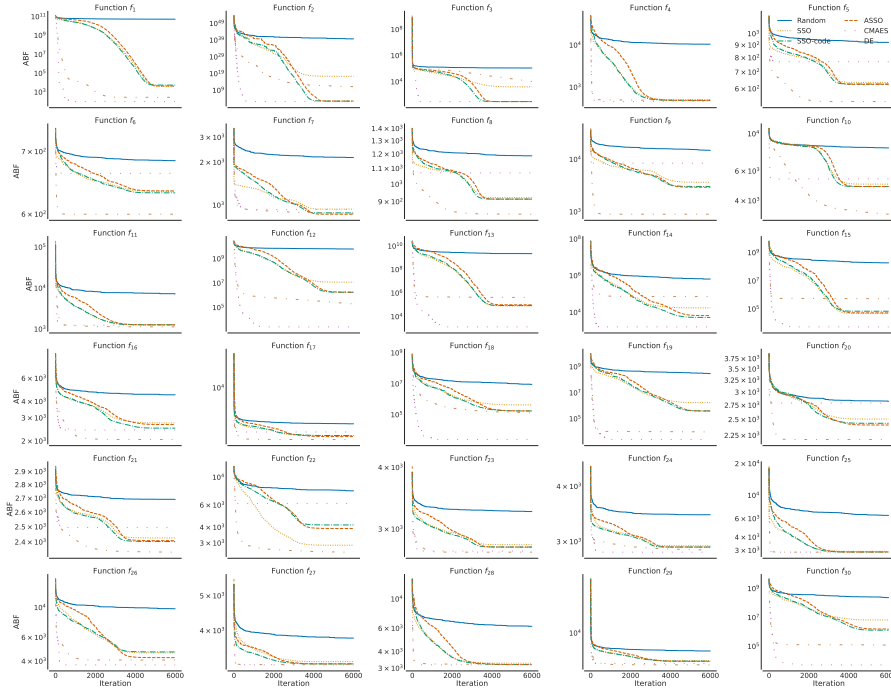
Figure 5: Comparison of the ABF obtained by the analyzed techniques for each function f_k (with $k = 1, \dots, 30$) and $D = 30$ The ABF was calculated by using the fitness values of the best individual over the 30 repetitions. Note that the y-axes are on a logarithmic scale.

depends on the distance between the lower bound and the origin. As shown in our experiments, under some specific circumstances, this factor can significantly affect the search capabilities of SSO, which was outperformed by a simple random search. If the term lb_j was inadvertently introduced, *all* the current literature on SSO contains results not reflecting the intended definition of the algorithm. On the other hand, if the authors meant to insert the term, the SSO algorithm cannot work in spaces that have lower bounds too far from 0 in all dimensions. We also compared the described SSO-based approaches against DE and CMAES for the optimization of the CEC 2017 benchmark functions [9], showing that all the SSO-based versions were outperformed by a simple DE version on almost all functions. These results highlight, once more, that SSO and similar algorithms do not give any particular advantages with respect to widespread and common metaheuristics. Considering all the results discussed in this work, we expect that more sophisticated metaheuristics, such as Linear population size reduction Successful History-based Adaptive DE (L-SHADE) [48, 50, 51, 61] that already showed their superior performance compared to DE, should outperform all the SSO-based approaches. Thus, based on the evidences of this work, we discourage the use of SSO by the scientific community. In particular, there is no theory supporting the convergence properties of SSO and its (supposed) superiority with respect to the existing metaheuristics. On the contrary, SSO is defined and implemented based on a wrong mathematical formulation, as discussed in the first part of this paper.

We conclude this paper with a general remark about the metaphor-based approach
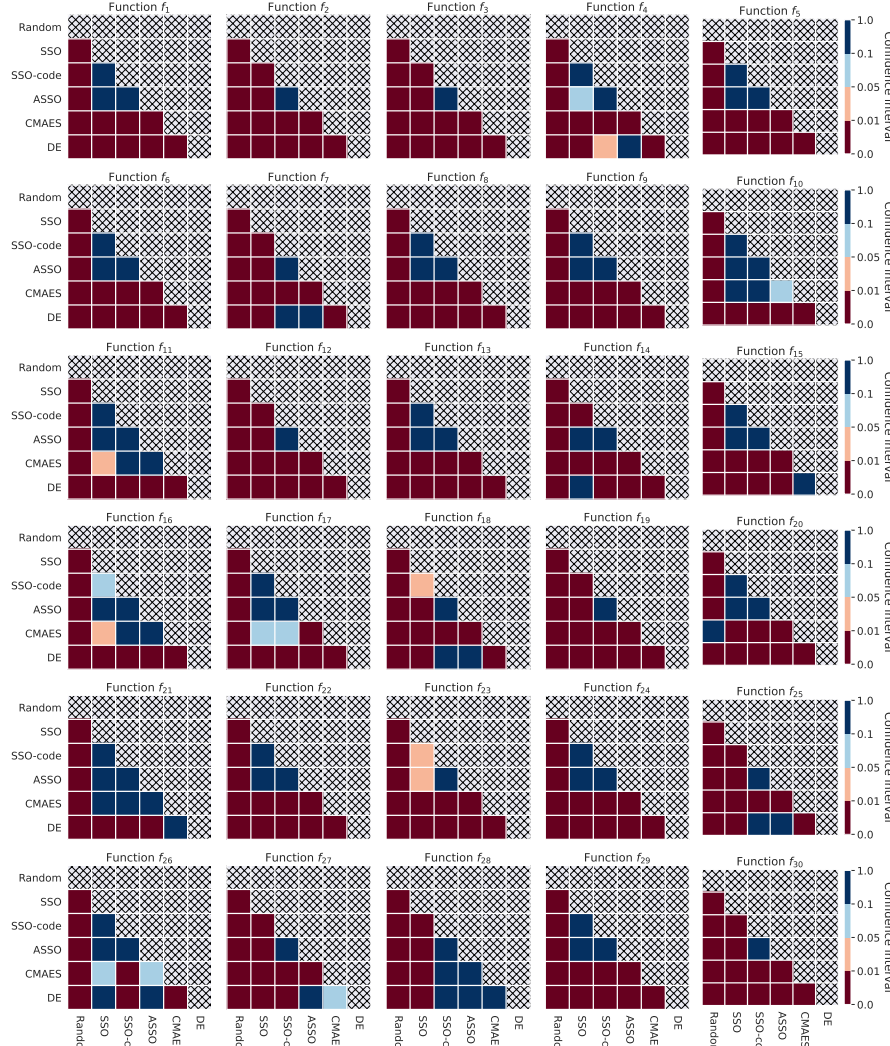
15

Figure 6: Heatmaps showing the p-values of the Mann–Whitney U test with the Bonferroni correction for each function f_k (with $k = 1, \ldots, 30$) and and $D = 30$. The confidence interval was divided into 4 levels indicating a strong statistical difference, statistical difference, weak statistical difference, and no statistical difference, respectively.

for metaheuristics. As mentioned in the Introduction, a lot of metaheuristic optimization algorithms have been proposed in the last years, most of them based on a particular natural process or animal behavior as a metaphor for the exploration and exploitation phases of the search space. As noted by [59], one of the likely causes of this phenomenon is the excessive focus on the *novelty* of such methods in part of the research community on metaheuristics. This research approach, however, has the downside of shadowing the true search components of an optimization algorithm with terms and concepts borrowed from the considered metaphor. Therefore, it can happen that a "novel" metaheuristic optimization technique turns out to be just another well-known algorithm under a

heavy disguise. This is the case, for example, of three other recent Swarm Intelligence algorithms, namely: the Grey Wolf Optimizer, the Firefly Optimization Algorithm, and the Bat Algorithm, which were shown by [17] to have strong similarities with Particle Swarm Optimization. While in this manuscript we showed and fixed the methodological and implementation flaws of SSO, we believe that a closer inspection of the algorithm's underlying metaphor would also highlight a strong resemblance to other established swarm algorithms.

Availability

All source code used for the tests is available on GitLab at the following address: `https://gitlab.com/andrea-tango/asso`.

Acknowledgment

References

[1] R. Abbassi, A. Abbassi, A. A. Heidari, and S. Mirjalili. An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. *Energy Convers. Manag.*, 179:362–372, 2019.

[2] L. Abualigah, M. Shehab, M. Alshinwan, and H. Alabool. Salp swarm algorithm: a comprehensive survey. *Neural. Comput. Appl.*, 32:11195—-11215, 2019.

[3] S. Ahmed, M. Mafarja, H. Faris, and I. Aljarah. Feature selection using salp swarm algorithm with chaos. In *Proc. 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (ISMSI)*, pages 65–69, 2018.

[4] M. A. Al-Qaness, A. A. Ewees, H. Fan, and M. Abd El Aziz. Optimization method for forecasting confirmed cases of COVID-19 in China. *J. Clin. Med.*, 9(3):674, 2020.

[5] A. Ala'a, A. A. Alsewari, H. S. Alamri, and K. Z. Zamli. Comprehensive review of the development of the harmony search algorithm and its applications. *IEEE Access*, 7:14233–14245, 2019.

[6] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang, and S. Mirjalili. Asynchronous accelerating multi-leader salp chains for feature selection. *Appl. Soft Comput.*, 71:964–979, 2018.

[7] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang, and S. Mirjalili. Asynchronous accelerating multi-leader salp chains for feature selection. *Appl. Soft Comput.*, 71:964–979, 2018.

[8] S. S. Alresheedi, S. Lu, M. Abd Elaziz, and A. A. Ewees. Improved multiobjective salp swarm optimization for virtual machine placement in cloud computing. *Hum.-centric Comput. Inf. Sci.*, 9(15), 2019.

[9] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization. Technical report, Nanyang Technological University, Singapore, and Jordan University of Science and Technology, Jordan, and Zhengzhou University, China, 2016.

[10] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[11] Z. Cheng and J. Wang. A new combined model based on multi-objective salp swarm optimization for wind speed forecasting. *Appl. Soft Comput.*, 92:106294, 2020.

[12] C. A. Coello Coello and N. C. Cortés. Solving multiobjective optimization problems using an artificial immune system. *Genet. Program. Evol. M.*, 6(2):163–190, 2005.

[13] Congress on Evolutionary Computation (CEC). Special Session & Competitions on Real-Parameter Single Objective Optimization. `http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2017/CEC2017.htm`, 2017. Online; Accessed on November 17, 2018.

[14] D. P. F. Cruz, R. D. Maia, and L. N. De Castro. A critical discussion into the core of swarm intelligence algorithms. *Evolutionary Intelligence*, 12(2):189–200, 2019.

[15] C. Doerr, F. Ye, S. van Rijn, H. Wang, and T. Bäck. Towards a theory-guided benchmarking suite for discrete black-box optimization heuristics: profiling (1+ λ) ea variants on onemax and leadingones. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 951–958, New York, NY, USA, 2018. ACM.

[16] M. Dorigo. Swarm intelligence: A few things you need to know if you want to publish in this journal, 2016.

[17] M. Dorigo. Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. In *Proc. 12th International Conference on Swarm Intelligence (ANTS)*, Lecture Notes in Computer Science, pages 121–133. Springer Nature, 2020.

[18] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Comput. Intell. M.*, 1(4):28–39, 2006.

[19] O. J. Dunn. Multiple comparisons among means. *J. Am. Stat. Assoc.*, 56(293):52–64, 1961.

[20] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proc. 6th International Symposium on Micro Machine and Human Science (MHS)*, pages 39–43. IEEE, 1995.

[21] S. Ekinci and B. Hekimoglu. Parameter optimization of power system stabilizer via salp swarm algorithm. In *Proc. 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, pages 143–147. IEEE, 2018.

[22] A. A. El-Fergany. Extracting optimal parameters of PEM fuel cells using salp swarm optimizer. *Renew. Energy*, 119:641–648, 2018.

[23] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita. An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowl-Based Syst.*, 154:43–67, 2018.

[24] S. Fong. Meta-zoo-heuristic algorithms. In *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, pages 3–8. IEEE, 2017.

[25] M. Gallagher. Towards improved benchmarking of black-box optimization algorithms using clustering problems. *Soft Comput.*, 20(10):3835–3849, 2016.

[26] C. García-Martínez, P. D. Gutiérrez, D. Molina, M. Lozano, and F. Herrera. Since cec 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. *Soft Computing*, 21(19):5573–5583, 2017.

[27] Z. W. Geem. Harmony search algorithm for solving sudoku. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 371–378. Springer, 2007.

[28] Genetic and Evolutionary Computation Conference (GECCO). GECCO Workshop on Real-Parameter Black-Box Optimization Benchmarking (BBOB). http://numbbo.github.io/workshops/BBOB-2018/, 2018. Online; Accessed on November 17, 2018.

[29] 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.

[30] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proc. International Conference on Evolutionary Computation (CEC)*, pages 312–317. IEEE, 1996.

[31] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[32] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[33] H. S. Hosseini. Problem solving by intelligent water drops. In *2007 IEEE congress on evolutionary computation*, pages 3226–3231. IEEE, 2007.

[34] A. G. Hussien, A. E. Hassanien, and E. H. Houssein. Swarming behaviour of salps algorithm for predicting chemical compound activities. In *Proc. 8th International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 315–320. IEEE, 2017.

[35] R. A. Ibrahim, A. A. Ewees, D. Oliva, M. Abd Elaziz, and S. Lu. Improved salp swarm algorithm based on particle swarm optimization for feature selection. *J. Ambient Intell. Humaniz. Comput.*, 10(8):3155–3169, 2019.

[36] V. Kansal and J. S. Dhillon. Emended salp swarm algorithm for multiobjective electric power dispatch problem. *Appl. Soft Comput.*, 90:106172, 2020.

[37] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artif. Intell. Rev.*, 42(1):21–57, 2014.

[38] A. V. Kononova, D. W. Corne, P. De Wilde, V. Shneer, and F. Caraffini. Structural bias in population-based algorithms. *Information Sciences*, 298:468–490, 2015.

[39] K. S. Lee and Z. W. Geem. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering*, 194(36-38):3902–3933, 2005.

[40] A. J. Lockett and R. Miikkulainen. A probabilistic reformulation of no free lunch: Continuous lunches are not free. *Evol. Comput.*, 25(3):503–528, 2017.

[41] W. G. Macready and D. H. Wolpert. What makes an optimization problem hard? *Complexity*, 1(5):40–46, 1996.

[42] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. of Math. Stat.*, pages 50–60, 1947.

[43] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.*, 114:163–191, 2017.

[44] S. Mirjalili, S. M. Mirjalili, and A. Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[45] P. Niu, S. Niu, L. Chang, et al. The defect of the grey wolf optimization algorithm and its verification method. *Knowledge-Based Systems*, 171:37–43, 2019.

[46] M. S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, and G. Pasi. Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization. *Swarm Evol. Comput.*, 39:70–85, 2018.

[47] M. S. Nobile, A. Tangherloni, L. Rundo, S. Spolaor, D. Besozzi, G. Mauri, and P. Cazzaniga. Computational intelligence for parameter estimation of biochemical systems. In *Proc. Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[48] A. P. Piotrowski. L-SHADE optimization algorithms with population-wide inertia. *Inf. Sci.*, 468:117–141, 2018.

[49] A. P. Piotrowski and J. J. Napiorkowski. Some metaheuristics should be simplified. *Information Sciences*, 427:32–62, 2018.

[50] A. P. Piotrowski and J. J. Napiorkowski. Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure? *Swarm Evol. Comput.*, 43:88–108, 2018.

[51] R. Polakova. L-SHADE with competing strategies applied to constrained optimization. In *proc. IEEE congress on evolutionary computation (CEC)*, pages 1683–1689. IEEE, 2017.

[52] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intell.*, 1(1):33–57, 2007.

[53] K. Price, R. M. Storn, and J. A. Lampinen. *Differential evolution: a practical approach to global optimization.* Springer Science & Business Media, 2006.

[54] M. H. Qais, H. M. Hasanien, and S. Alghuwainem. Enhanced salp swarm algorithm: Application to variable speed wind generators. *Eng. Appl. Artif. Intell.*, 80:82–96, 2019.

[55] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi. GSA: a gravitational search algorithm. *Inf. Sci.*, 179(13):2232–2248, 2009.

[56] G. I. Sayed, G. Khoriba, and M. H. Haggag. A novel chaotic salp swarm algorithm for global optimization and feature selection. *Appl. Intell.*, 48(10):3462–3481, 2018.

[57] H.-P. Schwefel. *Numerical optimization of computer models.* John Wiley & Sons, Inc., 1981.

[58] A. M. Shaheen and R. A. El-Sehiemy. A multiobjective salp optimization algorithm for techno-economic-based performance enhancement of distribution networks. *IEEE Syst. J.*, 15(1):1458–1466, 2020.

[59] K. Sörensen. Metaheuristics - the metaphor exposed. *Int. Trans. Oper. Res.*, 22(1):3–18, 2015.

[60] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.*, 11(4):341–359, 1997.

[61] R. Tanabe and A. S. Fukunaga. Improving the search performance of SHADE using linear population size reduction. In *prc. IEEE congress on evolutionary computation (CEC)*, pages 1658–1665. IEEE, 2014.

[62] A. Tangherloni, L. Rundo, and M. S. Nobile. Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems. In *Proc. Congress on Evolutionary Computation (CEC)*, pages 1940–1947. IEEE, 2017.

[63] A. Tangherloni, S. Spolaor, P. Cazzaniga, D. Besozzi, L. Rundo, G. Mauri, and M. S. Nobile. Biochemical parameter estimation vs. benchmark functions: a comparative study of optimization performance and representation design. *Appl. Soft Comput.*, 81:105494, 2019.

[64] F. Tian, H. Wei, X. Li, M. Lv, and P. Wang. An improved salp optimization algorithm inspired by quantum computing. In *Proc. 3rd International Conference on Advanced Algorithms and Control Engineering (ICAACE)*, volume 1570, pages 1–7. IOP Publishing, 2020.

[65] M. Tolba, H. Rezk, A. A. Z. Diab, and M. Al-Dhaifallah. A novel robust methodology based salp swarm algorithm for allocation and capacity of renewable distributed generators on distribution grids. *Energies*, 11(10):2556, 2018.

[66] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, and A. Zamuda. DISH-XX Solving CEC2020 Single Objective Bound Constrained Numerical optimization Benchmark. In *Proc. Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[67] C. C. Villalón, T. Stützle, and M. Dorigo. Cuckoo search≡$(\mu+\lambda)$–evolution strategy. *IRIDIA – Technical Report Series*, 6:1–15, 2021.

[68] C. L. Villalón, M. Dorigo, and T. Stützle. The intelligent water drops algorithm: why it cannot be considered a novel algorithm. *Swarm Intelligence*, 13(3):173–192, 2019.

[69] D. Weyland. A critical analysis of the harmony search algorithm—how not to solve sudoku. *Operations Research Perspectives*, 2:97–105, 2015.

[70] F. Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[71] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.*, 1(1):67–82, 1997.

[72] B. Yang, L. Zhong, X. Zhang, H. Shu, T. Yu, H. Li, L. Jiang, and L. Sun. Novel bio-inspired memetic salp swarm algorithm and application to MPPT for PV systems considering partial shading condition. *J. Clean. Prod.*, 215:1203–1222, 2019.

[73] W. Yang, J. Wang, H. Lu, T. Niu, and P. Du. Hybrid wind energy forecasting and analysis system based on divide and conquer scheme: A case study in China. *J. Clean. Prod.*, 222:942–959, 2019.

[74] X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Proc. Nature inspired cooperative strategies for optimization (NICSO)*, Studies in Computational Intelligence, pages 65–74. Springer, 2010.

[75] X.-S. Yang and S. Deb. Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4):330–343, 2010.

[76] X.-S. Yang and X. He. Firefly algorithm: recent advances and applications. *Int. J. Swarm Intell.*, 1(1):36–50, 2013.