A Hybrid Meta-Heuristic Algorithm for High Performance Computing

Elham Mahdipour, Mohammad Ghasemzadeh*

Computer Engineering Department, Yazd University, Yazd, Iran.
elham.mahdipour@stu.yazd.ac.ir
m.ghasemzadeh@yazd.ac.ir
*Corresponding author
Received: 2020-03-04
Revised: 2020-09-27
Accepted: 2020-09-28

Abstract
Regarding optimization problems, there is a high demand for high-performance algorithms that can process the problem solution-space efficiently and find the best ones quite quickly. An approach to get this target is based on using swarm intelligence algorithms; these algorithms apply a population of simple agents to communicate locally with one another and with their surroundings. In this paper, we propose a novel approach based on combining the characteristics of the two algorithms: Cat Swarm Optimization (CSO) and the Shuffled Frog Leaping Algorithm (SFLA). The experimental results show the convergence ratio of our hybrid SFLA-CSO algorithm is seven times higher than that of CSO and five times higher than the convergence ratio of the standard SFLA algorithm. The obtained results also revealed that the hybrid method speeds up the convergence significantly, and reduces the error rate. We compared the proposed hybrid algorithm against the famous relevant algorithms PSO, ACO, ABC, GA, and SA; the results are valuable and promising.

Keywords
Cat swarm optimization, Convergence rate, Shuffled frog leaping algorithm, Swarm intelligence.

1. Introduction

There are many optimization problems in science and engineering, which are naturally more complicated and challenging to be solved by conventional optimization methods; Concerning these problems, a practical approach is to use evolutionary methods [1, 2]. Most of these problems in the real world are dynamic; it means that their optima may change over time. Therefore, algorithms to solve these problems must be well adapted to their conditions in such a way that they should be able to track the optima during evolution [3]. The high-performance algorithms are the most efficient methods known for representing, and analysing such systems [4].
High-performance computing means cloud computing that requires extensive equipment. High-performance algorithms can perform these complex calculations. Our proposed method has the potential to parallelize and can be used in high-performance environments. High performance in this paper means providing a faster algorithm than other algorithms, and we evaluated this in terms of the average number of iterations required to achieve the optimal solution.
In optimization problems, we need to use high-performance algorithms such as swarm intelligence algorithms. These algorithms employ a population of simple agents to communicate locally with one another and with their surroundings. In this regard, we present a hybrid method for improving the convergence rate of the shuffled frog leaping algorithm (SFLA) and cat swarm optimization (CSO) algorithms. We demonstrate the hybrid proposed method is faster than standard SFLA and CSO algorithms. By reaching the global optimum or reasonable local solutions in a practical time, researchers introduce new optimization methods.
Regarding the primary shortcoming of the previous studies, it worth noting that despite a faster convergence compared with Particle Swarm Optimization (PSO) algorithm, application of CSO is highly limited by the drawback of "premature convergence," that is, the possibility of trapping in local optimum when dealing optimization problems with a large number of local extreme values [5]. Because of the weaknesses of the SFLA for optimizing some functions such as a low optimization precision, a slow speed, and trapping into the local optimum easily, a hybrid shuffled frog leaping algorithm and cat swarm optimization (SFLA-CSO) is proposed in this research [6].
The gap this research is trying to fill is obtaining faster convergence and a higher speed in finding proper solutions. This hybrid view is based on the premise that if a frog uses a cat tracing mode and speed when hunting, it can catch its prey faster. Almost all meta-heuristic algorithms can be used offline, and we are continually witnessing new hybrid algorithms that show better performance in speed and convergence. We have also proposed a solution that combines two optimization algorithms to find the global optimum much faster. Since the SFLA has the potential to be parallelized, it is possible to develop the proposed algorithm for online and even real-time algorithms in future work.

The paper is structured as follows: In Section 2, we introduce the Shuffled Frog Leaping Algorithm (SFLA), Cat swarm optimization (CSO) algorithm, and related works; Section 3 proposes the hybrid method for improving the convergence speed of the two SFLA and CSO algorithms, that we called Hybrid SFLA-CSO; Section 4 reports the experimental results obtained from testing the proposed method and the standard SFLA, CSO, and other algorithms on benchmark functions; finally, Section 5 is the conclusion part and suggests further research.

2. SFLA and CSO Algorithms

In this section, we describe SFLA and CSO algorithm, then we consider related works.

2.1. Shuffled Frog leaping Algorithm

Eusuff and Lansey presented SFLA to be used in the optimization of water networks [7]. The chief idea of this algorithm is to obtain a search method for solving complex optimization problems without using mathematical relations. The SFLA inspired the lives of the frogs, and each frog illustrates a possible solution to the optimization problem. This algorithm attempts to improve the worst-case in three random jumps, so the jumper frog algorithm is named. In this algorithm, the memeplex is defined as a group of frogs. In the first step, in each memeplex, the worst solution $X_{w,k}$ attempts to improve its position by exchanging its position information relative to the best answer in the same sample $X_{b,k}$; where k is current memeplex. Using the (1) and (2), this exchange of information takes place where a rand is a random number between [0, 1].

$$Si = rand.(X_{b,k} - X_{w,k}) \tag{1}$$

$$X_{w,k} = X_{w,k} + Si \tag{2}$$

In case the new position is not better than before, in the second step, the exchange of information between the memeplex takes place. The number of frogs in each memeplex is equal to n. The global best frog (X_g) is used to improve the position, and in (1) term, X_g is used instead of $X_{b,k}$. Further still, in case no result is obtained, the researchers create a random number in the range of domain solution and replace with the worst frog in the memeplex. Fig. 1 shows the division frogs in each memeplex.

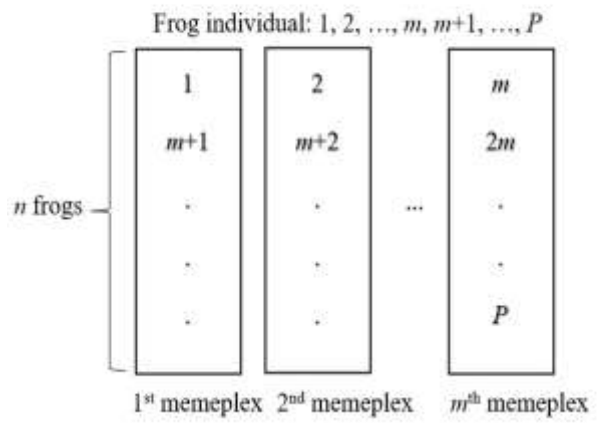

Fig.1. Division frogs in each memeplex [7]

Algorithm 1. Shuffled Frog Leaping Algorithm

```
1.  Input: number of the primary population ($P$), the number of the memeplex ($m$), and the number of iterations in each memeplex ($It$), Maximum number of global iteration ($Epoch$).
2.  Create a $P$ random first generation;
3.  Calculate the fitness function for $P$ population;
4.  Sort the frog population based on the fitness;
5.  Define the terminal condition, such as find the global optimum, or arrive at the desired iteration, or arrive at error tolerance (0.01).
6.  while ($Epoch$ or Terminal Condition==True)
7.    Divide $P$ into $m$ memeplex such that $P = m.n$; (e.g., $m$ memeplex with $n$ frogs).
8.    while($i<m$)
9.      while($j<It$)
10.       Compute $X_b$ (best frog in the memeplex), $X_w$ (worst frog in the memeplex), and $X_g$ (global best frog between all population) for current memeplex ($m$).
11.       Compute the $Si$ and set $X_w$, according to Eq.(1) and (2), such that $S_{min} < Si < S_{max}$. $S_{max}$ is the maximum value of the solution domain ($D$) and $S_{min}$ is the minimum value of the solution domain.
12.       $flag1 \leftarrow False$
13.       if the fitness of the new $X_w$ better than the old $X_w$:
14.         Replace a new frog with the worst frog in memeplex.
15.         $flag1 \leftarrow True$
16.       if $flag1$==False:
17.         $flag2 \leftarrow False$
18.         use $X_g$ instead of $X_b$ in equation (1).
19.         if the fitness of the new frog position better than the old position:
20.           Replace a new frog with the worst frog in memeplex.
21.           $flag2 \leftarrow True$
22.       if $flag1$==False and $flag2$==False:
23.         Create a new random frog
24.         Replace new frog with the worst frog in the memeplex
25.     end while $j$
26.   end while $i$
27.   Combine all memeplexes and select $P$ best particles to create a new population for the next generation
28. end while $Epoch$
29. Output: Global Optimum Solution
```

We describe the steps of the shuffled frog leaping algorithm in Algorithm 1. Let us assume that the maximum desired iteration is 5000, and error tolerance is 0.001 for all algorithms in this paper.

2.2. Cat Swarm Optimization

In this subsection, we introduce the CSO [8] algorithm. In CSO, there are N cats, and each cat has m dimensions in the algorithm, and each dimension has a velocity. This algorithm is inspired by the behavior of cats. There are two modes in the behavior of cats so-called tracing mode and seeking mode. Each cat with MR probability is put in tracing mode, and 1-MR probability put into seeking mode. Each cat has a flag that shows; the cat is in which mode.

In seeking mode, there are four essential factors as follows: Seeking Memory Pool (SMP), Seeking Range of the Dimension (SRD), Counts of Dimension to Change (CDC), and Self-Position Considering (SPC).

Then, given these factors, when the cat is in seeking mode, we act as follows [8]:

- **Step 1:** Create j copies of the current position of Cat_k. The amount of j is considered to be the Seeking Memory Pool (j=SMP). If the selected cat's SPC value is true, the value of j is set to SMP-1, and then maintains the position of the current cat as one of the candidates.
- **Step 2:** For each copy, under the CDC, randomly plus or minus SRD percent of the current values and supersede the previous ones.
- **Step 3:** Compute the fitness values (FS) for each of the candidate points.
- **Step 4:** If all the FS are not equal, compute the probability of each of the candidates using (3). Otherwise, the probability of selecting all the points of the candidate one is considered.

$$P = \frac{|FS_i - FS_b|}{FS_{max} - FS_{min}} \quad 0<i<j \tag{3}$$

If the aim of the fitness function is maximized, FS_b in (3) is equal with $FS_{\min}$ else if the goal is minimized,s FS_b is equal with $FS_{\max}$.

- **Step 5:** Select the best candidate to move and replace with the Cat_k position.

We consider the tracing mode for the CSO algorithm. Considering this case, where the cat moves according to its own velocities for each dimension. The tracing mode process expressed as follows:

- **Step 1:** According to (4), every dimension ($V_{k,d}$) is updated.

$$V_{k,d} = V_{k,d} + r_1.c_1.(X_{best,d} - X_{k,d}) \tag{4}$$

where d=1, 2, …, M and $X_{best,d}$ represents the position of the best cat with the best fitness and $X_{k,d}$ is the position of the Cat_k. c_1 is a constant and r_1 is a random number in the interval [0, 1].

- **Step 2:** Check the velocity within the defined range. Otherwise, the value is set to the maximum velocity.
- **Step 3:** Update the Cat_k by (5).

$$Xk,d = Xk,d + Vk,d \tag{5}$$

2.3. Related Works

In this section, we review the related works of SFLA and CSO in recent years.

Cai et al. proposed a dynamic shuffled frog-leaping algorithm (DSFLA) to minimize make span [9]. The dynamic search process is executed in each memeplex with at least two improved solutions. Global search and dynamic multiple neighbourhood searches are applied, in which neighbourhood structure is chosen based on its optimization effect. A new destruction-construction process is hybridized with DSFLA and population shuffling is done when shuffling condition is met.

Tang et al. proposed a discrete shuffled frog-leaping algorithm to solve the influence maximization problem in a more efficient way [10]. Novel encoding mechanism and discrete evolutionary rules are conceived based on the network topology structure for the virtual frog population. To facilitate the global exploratory solution, a novel local exploitation mechanism combining deterministic and random walk strategies is put forward to improve the suboptimal meme of each memeplex in the frog population.

Kaveh et al. [11] presented an efficient hybrid optimization algorithm based on an invasive weed optimization algorithm and SFLA for optimum design of trusses. Their algorithm converges to better, or at least the same solutions than some other methods, while the number of structural analyses was reduced. They compared the proposed method against other metaheuristic optimization methods.

Dash et al. [12] proposed an SFLA technique for feature selection of high dimensional data. Thus, they suggested a binary SFLA for gene selection in gene expression data. Performance of proposed evolutionary framework compared with three other learning approaches such as PSO, DE and GA using KNN, ANN, and SVM, respectively. The classification result showed that binary SFLA was stable in most cases.

Dash [13] used an improved shuffled frog leaping algorithm to approximate the unrevealed parameters of the neural network. Sharma et al. [14] presented an SFLA with improved detection, optimal locations for equipment to decrease the noise level in the multi noise plant.

Daoden et al. [15] introduced the SFLA, a meta-heuristic approach for optimization, hybrid with the bottom left fill algorithm as an approach for optimizing the arrangement of the 2D shapes in infinite space.

Kaur et al. [16] presented an augmented shuffled frog leaping algorithm based on the technique for resource provisioning and work-flow scheduling in the Infrastructure as a service cloud environment.

Flexible job-shop scheduling problem (FJSP) with the minimization of workload balance and total energy consumption was considered and the conflict between two objectives was analyzed [17]. They proposed an SFLA based on a three-string coding approach. Their experiments depicted the conflicting between two objectives of FJSP and the promising advantages of SFLA on the considered FJSP.

The activity-based costing system allocates the overhead costs more accurately to cost objects than standard costing systems. The SFLA, a meta-heuristic method is applied in selecting optimal representative cost drivers [18] for discovering optimal solutions. The objective function of the algorithm is the cost-saving from the information gathering cost of deleted cost drivers minus the loss of accuracy cost. The experimental results showed that the SFLA can effectively find the optimal cost driver composition with the optimal objective function value.

Villa et al. [19] described a method to solve optimization problems based on the binary cat swarm optimization (BCSO) algorithm to optimize the trajectory of a mobile unicycle autonomous robot, to follow an established trajectory with the lowest possible margin of error. Simulation results show that the proposed method achieves good results.

Murtza et al. [20] proposed a multi-objective integer CSO algorithm. Their approach comprised of the concepts of rounding the floating-point to the nearest integer numbers and the probabilistic updating technique. It used the idea of Pareto dominance for finding the non-dominated solutions and an external archive for storing these solutions. The simulation results argued that the proposed approach can be a better candidate for solving the integer multi-objective problems.

Soto et al. [21] presented a Binary CSO (BCSO) for solving the Manufacturing Cell Design Problem. Their goal was to identify a cellular organization in such a way that the transportation of the different parts between cells was minimized. The experimental results for both typical BCSO and autonomous search BCSO reached all global optimums.

Pappula et al. [22] presented a typical mutation strategy-based cat swarm optimization that features useful global search capabilities with accelerating convergence speed. A compact cat swarm optimization scheme proposed [23], designed to solve application domains plagued with limited memory and less-computation power.

Thomas et al. [24] proposed a new simulation-optimization model for aquifer parameter estimation by coupling the radial point collocation meshfree method with CSO. Skoullis et al. [25] presented the application of a hybrid CSO based on an algorithm for solving the school timetabling problem. Chen et al. [26] established a good placement optimization method using an analytical formula based objective function and CSO algorithm.

Recently, Kumar et al. [27], proposed a scheduling workflow application to optimize workflow schedule length also known as makespan. Scientific workflows of different sizes were evaluated with the proposed binary CSO algorithm using WorkflowSim and results were compared with state-of-the-art algorithms. The results indicated an improvement in the performance by minimizing the makespan.

Nie et al. [5] proposed a Chaos Quantum-behaved Cat Swarm Optimization (CQCSO) algorithm. They introduced a tent map for jumping out of local optimum in the CQCSO algorithm. The experimental results demonstrated the efficiency of the method. A CSO-based adaptive algorithm and its revision to modify the performance of a channel equalizer were proposed by Diana et al. [28]. The data were transmitted through diverse channel conditions. A linear transversal filter was used as a channel and equalizer model. The equalizer coefficients were trained by the proposed simplified cat swarm optimization algorithm to discover the estimated digital training data.

Furthermore, Wang et al. [29] suggested a new facial emotion recognition method. In the first step, they extracted wavelet coefficients from facial images using a discrete wavelet transform. The second step reduced the features by the principal component analysis. The third step applied a single-hidden-layer neural network classifier. Finally, and most importantly, they proposed the cat swarm optimization to train the weights and biases of the classifier.

Kencana et al. [30] considered the accuracy of the CSO algorithm in classifying small loan performance that was approved by Bank Rakyat Indonesia, one of several public banks in Indonesia. Also, a multi-objective quality of service model to address customer's profit based on run time and execute cost criteria was proposed [31]. To solve the model, they were presented a cloud scalable multi-objective CSO based on Simulated Annealing (SA) algorithm. In their method, the Taguchi used an orthogonal approach to enhance the SA and incorporated into the local search of the proposed algorithm to increase its exploration capability. Dhaliwal et al. [32] recommended an integrated optimization technique that combined the features of the CSO algorithm with the traditional differential evolution algorithm and applied it for the optimal design of digital infinite impulse response filters.

3. Hybrid SFLA-CSO proposed method

In this section, the SFLA and CSO were combined to improve the convergence rate of both algorithms. In the SFLA algorithm, frogs are moved only based on the position information. In the CSO algorithm, the cats move in tracing mode based on position and velocity. A clear view is that in Particle Swarm Optimization (PSO) and CSO algorithms, the particle speed has a significant role in convergence. The PSO and CSO algorithm escapes from local optimizations using speed. Also, increasing the speed can lead to early convergence; however, with the correct set of the initial parameters required in each algorithm, global optimization can be achieved. In our proposed algorithm, we define a speed dimension for each frog to increase the convergence rate of the SFLA.

Accordingly, as the cat in the tracing mode first evaluates all his prey, then selects the best bait; in the proposed algorithm, each frog can use the velocity dimension to evaluate different positions in several directions and then choose the best position. This makes each frog find the best position in the current memeplex, and ultimately increase the speed of convergence. Similarly, by controlling and setting the initial parameters of the proposed algorithm, we prevent early convergence. Of course, setting these initial parameters is determined by the type of problem and the objective function.

The hybrid SFLA-CSO algorithm for each frog considers a vector of velocity, and we propose a new formula for the frog's displacement using the formulas of the cat tracing mode and the frog displacement formulas. For this purpose, we considered a velocity dimension for each frog in the SFLA algorithm when generating a population. We assumed the frogs have a velocity vector and their position and used the tracing mode in the CSO algorithm to update the data. Therefore, we modify the (1) and (2) of the SFLA algorithm as follows:

$$V_{w,k} = V_{w,k} + r_1.c_1.\left(X_{b,k} - X_{w,k}\right) \tag{6}$$

$$X_{w,k} = X_{w,k} + V_{w,k} \tag{7}$$

where k=1, 2, …, m and c_1 is a constant (c_1 = 2), r_1 is a random number in the interval [0, 1], $X_{b,k}$ is the best local frog (the best frog in the memeplex) in dimension k and $X_{w,k}$ is the worst frog in the memeplex in dimension k. After updating the worst frog, the amount of fitness was calculated, and in case the new fitness value does not give better than the old fitness value, we would use the X_g (global best frog) instead of $X_{b,k}$ using (8) and (9).

$$V_{w,k} = V_{w,k} + r_1.c_1.\left(X_{g,k} - X_{w,k}\right) \tag{8}$$

$$X_{w,k} = X_{w,k} + V_{w,k} \tag{9}$$

Once again, the particle's fitness is calculated, and if the superiority is not achieved yet, (10) will be used.

$$X_{w,k} = X_{w,k}.Rand(2) \tag{10}$$

where *Rand* (2) is a random number between [0, 2]. The reason for using a random number between zero and two is to make it possible to jump to the points farther or closer to the global optimum. This is one way to escape from the local optimum and increase exploration.
Algorithm 2 describes the steps of the Hybrid SFLA-CSO algorithm.
We obtained the mentioned improvements by imposing almost no cost to the basic SFLA; because the main operation of the SFLA is under jumping, and we have only added a formula for calculating the speed with $O(1)$ running time, which does not change the main time complexity of the algorithm.

4. Experimental Results

We used complex mathematical functions to test the performance of our proposed method. Ten numeric optimization problems were chosen to compare the relative performance of the Hybrid SFLA-CSO algorithm to SFLA, CSO, and others. These functions were standardized benchmark test functions with many local minima, bowl-shaped, plate-shaped, valley-shaped, and all were minimization problems. We assumed the dimension of all functions equals 30 (n = 30). *Table I* shows the mathematic benchmark functions [33]. Correspondingly, *Table II* shows the initial parameters for each algorithm. Let us assume that the maximum desired iteration (*Epoch*) is 5000, and error tolerance is 0.001 for all algorithms in this paper.
We implemented all of the procedures in MATLAB, on a microcomputer system with core-i7H CPU and 32 GB RAM. Comparing the performance of a hybrid algorithm with its fundamental ingredient methods is commonly done in similar research works.

Algorithm 2. Hybrid SFLA-CSO Algorithm

```
1.  Input: Initialize input parameters such as memeplex numbers (m), number of jumps in each
    memeplex (it), number of frogs in each memeplex (n), number of populations (P) that P = m.n.
    Also, we assume X is a frog in the memeplex, X_b is the best frog in memeplex and X_g is the
    global best of frogs in population. As well as X_{b,k} is the best frog in the kth memeplex;
    and X_{w,k} is the worst frog in kth memeplex, Maximum number of global iteration (Epoch).
2.  Create a random population and compute fitness.
3.  Sort the population based on fitness.
4.  Determine the global best frog.
5.  Define the terminal condition, such as find the global optimum, or arrive at desired
    iteration, or arrive at error tolerance.
6.  while (Epoch or Terminal Condition==True)
7.      Divide P into m memeplex such that P = m.n.; (i.e. n group with m frogs).
8.      while(i<m)
9.          while(j<It)
10.             Sort each memeplex.
11.             Determine the best frog and velocity in each memeplex ($X_{b,k}$; $V_{b,k}$)
12.             Determine the worst frog and velocity in each memeplex ($X_{w,k}$; $V_{w,k}$)
13.             Compute the $V_{w,k}$ and set $X_{w,k}$ according to Eq. (6) and (7) and save in
                TeV and TeX temporal variables for velocity and position, respectively.
14.             $FS_{new} \leftarrow Fitness(TeX)$
15.             $flag1 \leftarrow False$
16.             if new fitness better than old fitness:
17.                 Replace a new frog with the worst frog in memeplex.
18.                 $flag1 \leftarrow True$
19.             if flag1==False:
20.                 $flag2 \leftarrow False$
21.                 Compute the $V_{w,k}$ and set $X_{w,k}$ according to equations (8) and (9).
22.                 $FS_{new} \leftarrow Fitness(TeX)$
23.                 if the fitness of the new frog position better than the old position:
24.                     Replace a new frog with the worst frog in memeplex.
25.                     $flag2 \leftarrow True$
26.             if flag1==False and flag2==False:
27.                 Create a random frog ($X_{w,k}$) by Eq. (10)
28.                 Replace a new frog with the worst frog in memeplex.
29.         end while j
30.     end while i
31.     Combine all memeplexes and select P best particles to create a new population for the
        next generation.
32. end while Epoch
33. Output: Global Optimum Solution
```

Table I. Benchmark Functions

Function	Formula	
Rastrigin	$F_1(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\,cos(\,2\pi x_i)),\ \ x_i \in [-5.12,5.12]$	(11)
Griewangk	$F_2(x) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod cos(\frac{x_i}{\sqrt{i}}) + 1, \qquad x_i \in [-600,600]$	(12)
Sphere	$F_3(x) = \sum_{i=1}^{n} x_i^2, \qquad x_i \in [-5.12,5.12]$	(13)
Ackley	$F_4(x) = 20 + e - 20e^{-0.2\left[\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right]} - e^{\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)}, \quad x_i \in [-32.768,32.768]$	(14)
Schwefel's Double Sum	$F_5(x) = \sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2, \qquad x_i \in [-100,100]$	(15)
Step	$F_6(x) = 6n + \sum_{i=1}^{n}\lfloor x_i \rfloor, \qquad x_i \in [-5.12,5.12]$	(16)
Rosenbrock	$F_7(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \qquad x_i \in [-5, 10]$	(17)
Sum Squares	$F_8(x) = \sum_{i=1}^{n} i x_i^2 \qquad x_i \in [-10, 10]$	(18)
Zakharov	$F_9(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5\ i\ x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5\ i\ x_i\right)^4 \qquad x_i \in [-5, 10]$	(19)
Schwefel	$F_{10}(x) = 418.9829n - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert}) \qquad x_i \in [-500, 500]$	(20)

We compare the proposed method (SFLA-CSO) with other meta-heuristic and evolutionary algorithms, such as CSO, SFLA, Genetic Algorithm (GA), PSO, Simulated Annealing (SA), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC). *Table II* shows the parameter setting of these algorithms. *Table III* to *Table VI* shows the results of Hybrid SFLA-CSO in comparison with SFLA and CSO and other algorithms, respectively. Each examination runs 20 different times, and the average value was reported.

We say that the algorithm is convergent if it satisfies the equation (21):

$$\left|\frac{Fitness_{GlobalOptimum} - AvgFitness_t}{Fitness_{GlobalOptimum} - AvgFitness_{t-1}}\right| < Thr \qquad (21)$$

Fitness$_{GlobalOptimum}$ is the fitness value of the global optimum that found so far. *AvgFitness*$_t$ is the average of the best fitness until *t* iteration. *AvgFitness*$_{t-1}$ is the average of the best fitness until *t*-1 iteration. We set the threshold *Thr* to 0.001.

Based on the experimental results, other algorithms also require more iterations to reach the desired convergence rate. *Table III* shows the mean iteration of each algorithm for 20 different runs.

Also, we record the best error of each run, and after 20 runs, we compute the average results; you can see these results in the Mean Error column of *Table III*.

Fig. 2 shows the average of the run time after 20 runs. Fig. 3 shows the average of the standard deviation after 20 runs.

To compare the iteration number of each algorithm with another and to evaluate convergence speed, first of all, compute the average of iterations after 20 runs. Then, for each of the algorithms, we calculate the average of the iterations for all benchmark functions. Column 2 of *Table VII* shows the sum the average of iteration for each algorithm. Column 3 of *Table VII* shows the average of the mean iteration for each algorithm. Therefore, we compute the average of the convergence speed of each algorithm based on the iteration number. The following equation (i.e. (22)) shows how doing this for the SFLA algorithm. We define an Avg_Iteration variable for the mean iteration after 20 runs.

$$Mean\big(Avg_{Iteration(i)}\big) = \frac{\sum_{j=1}^{10} Avg_{it_i}(F_j(x))}{10} \quad S.t.\, i = \ SF\ LA, CSO, SFLA - CSO, GA, P\ SO, SA, ACO, ABC \qquad (22)$$

where *j* in F_j(x) is F_1(x) to F_{10}(x) of benchmark functions. *Table VII* shows the calculation results of this mean. To compare the mean number of iterations and the convergence speed of the proposed algorithm with other algorithms; we use the results of *Table VII* to calculate the number of iterations of the SFLA-CSO algorithm to other algorithms.

Table II. Set values of parameters for each algorithm

Type of Algorithm	Parameter	Value	Type of Algorithm	Parameter	Value
SFLA	*P* (population)	30	PSO	Swarms (population)	100
	it (iteration)	3		Particles (Dimension)	30
	m (memeplex)	6		c_1	2
	n (frogs)	5		c_2	2
	Dimension	30		r_1	[0,1]
	Epoch (Max iteration)	5000		Max iteration	5000
CSO	SMP	5	SFLA-CSO	*P* (population)	30
	SRD	0.2		*it* (iteration)	3
	CDC	0.8		*m* (memeplex)	6
	SPC	0.2		*n* (frogs)	5
	MR	0.2		c_1	2
	c_1	2		r_1	[0,1]
	r_1	[0,1]		Max Velocity	600
	Dimension	30		Dimension	30
	Max iteration	5000		Max iteration	5000
GA	Population	100	ACO	Population	100
	Genes (Dimension)	30		Dimension	30
	Crossover Rate	0.7		Probability threshold for maximum trail	0.9
	Mutation Rate	0.1		Local search probability	0.01
	Elitism Selection	25%		Evaporation Rate	0.01
	Max iteration	5000		Max iteration	5000
SA	Population	100	ABC	Population	100
	Dimension	30		Dimension	30
	Temperature	[1, 4]		Sites selected number for neighbourhood search	10
	Decrease step of temperature	0.001		Number of bees recruited for best sites	5
	Max iteration	5000		Max iteration	5000

Table III. Performance results of Hybrid SFLA-CSO and other algorithms for bowl-shaped benchmark functions

Function	Algorithm	Mean Error	Mean STD	Mean Run Time (Seconds)	Mean Iteration of 20 runs
Sphere	SFLA	0.0008	0.000313	0.3222	125
	CSO	0.0027	1.4206	54.9296	154
	SFLA-CSO	0.0001	0.000314	0.5335	26
	GA	0.0	0.000276	0.1816	55
	PSO	0.0944	0.000732	1.5688	805
	SA	0.0926	0.000277	0.362	955
	ACO	0.0624	0.0	0.0219	836
	ABC	0.0009	0.0000009	5.9118	2508
Schwefel's Double Sum	SFLA	0.0064	0.000277	5.0727	345
	CSO	0.0086	4.18	240.0825	260
	SFLA-CSO	0.0002	0.000314	6.8133	80
	GA	0.0	0.000024	8.4684	13
	PSO	0.0865	4.813	4.604	1964
	SA	0.0876	3.81	5.5892	1746
	ACO	16.52	2.681	30.6205	5000
	ABC	56.30	2.575	35.2605	5000
Sum Squares	SFLA	0.000045	0.000303	4.2576	585
	CSO	0.09816	3.912	52.2731	2506
	SFLA-CSO	0.0	0.000284	8.3036	496
	GA	0.000485	0.000149	1.3565	730
	PSO	0.09741	7.7993	1.4285	3457
	SA	0.089453	4.87245	2.9737	4002
	ACO	0.0	0.0	2.5535	530
	ABC	0.0	0.0	6.8997	1157

Table IV. Performance results of Hybrid SFLA-CSO and other algorithms for many local minima benchmark functions

Function	Algorithm	Mean Error	Mean STD	Mean Run Time (Seconds)	Mean Iteration of 20 runs
Rastrigin	SFLA	0.0	0.000327	3.4351	350
	CSO	0.0065	7.0024	59.9939	400
	SFLA-CSO	0.0	0.000286	3.7337	32
	GA	0.0	0.000095	1.1926	1211
	PSO	0.0978	3.9559	3.4435	2989
	SA	0.0935	6.9923	87.5547	2865
	ACO	0.0004	0.0	2.0998	756
	ABC	0.0031	0.0	4.6222	4674
Griewangk	SFLA	0.0001	0.000292	3.6062	168
	CSO	0.0001	7.2967	46.2424	140
	SFLA-CSO	0.0	0.000345	5.5911	43
	GA	0.0	0.000104	0.7677	16
	PSO	0.0822	8.1049	4.8779	2004
	SA	0.0986	3.0215	7.6174	2105
	ACO	0.0860	0.0	2.3689	605
	ABC	0.0055	0.0	5.8076	3750
Ackley	SFLA	0.0009	0.0475	5.7434	285
	CSO	0.0073	0.0898	47.2429	300
	SFLA-CSO	0.0004	0.0135	7.6286	60
	GA	0.0	0.1052	34.7092	25
	PSO	0.0997	0.0357	12.1546	4909
	SA	0.0984	0.0672	11.5187	2500
	ACO	0.7300	0.0691	6.7928	1008
	ABC	0.0013	0.0454	10.4609	1584
Schwefel	SFLA	0.0160	4.611997	5.3815	5000
	CSO	0.0820	4.501198	46.2415	5000
	SFLA-CSO	0.0	0.278	1.4081	151
	GA	0.0427	5.5479	31.006	5000
	PSO	0.02348	1.013	3.0714	5000
	SA	0.1669	3.43369	5.2104	5000
	ACO	0.0	5.41	6.3193	3500
	ABC	0.0	4.567	4.38	1200

Table V. Performance results of Hybrid SFLA-CSO and other algorithms for plate-shaped benchmark functions

Function	Algorithm	Mean Error	Mean STD	Mean Run Time (Seconds)	Mean Iteration of 20 runs
Step	SFLA	0.0265	0.0	3.5585	400
	CSO	0.0	6.4195	49.6351	6
	SFLA-CSO	0.0	0.8837	0.0095	4
	GA	0.0962	5.4895	13.0209	5000
	PSO	0.0	1.80299	5.1798	10
	SA	0.0739	5.0357	46.8004	3100
	ACO	0.0982	2.2278	0.0211	1640
	ABC	0.0868	4.66	6.2012	2860
Zakharov	SFLA	0.000015	0.000304	4.4623	615
	CSO	0.09862	4.5557	258.12	3100
	SFLA-CSO	0.0	0.001627	6.8147	236
	GA	0.000676	0.000081	2.7063	718
	PSO	0.094173	2.99	2.081	4800
	SA	0.9764	3.15	77.4449	5000
	ACO	0.8591	1.822	6.0042	5000
	ABC	0.9715	4.3093	12.0847	5000

Table VI. Performance results of Hybrid SFLA-CSO and other algorithms for valley-shaped benchmark function

Function	Algorithm	Mean Error	Mean STD	Mean Run Time (Seconds)	Mean Iteration of 20 runs
Rosenbrock	SFLA	28.2526	0.1585	7.1022	2500
	CSO	75.641	1.05	53.6994	2286
	SFLA-CSO	24.5677	0.0685	9.2538	754
	GA	28.8617	0.000608	39.6623	4800
	PSO	42.3203	4.4186	1.3945	5000
	SA	52.287	1.058	3.4194	5000
	ACO	25.3471	1.75439	6.0955	4500
	ABC	25.2371	0.1546	11.9473	4750

The (23) shows how this ratio is calculated.

$$R = \frac{Mean(Avg_{Iteration(i)})}{Mean(Avg_{Iteration(SFLA-CSO)})} \quad s.t. \quad i = SFLA, CSO, GA, PSO, SA, ACO, ABC \tag{23}$$

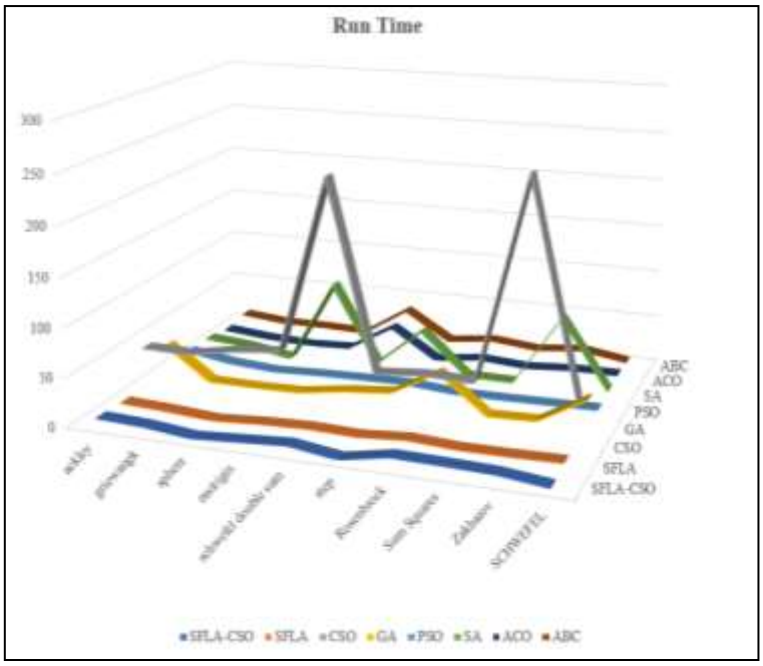

Fig. 2. Run Time of Hybrid SFLA-CSO, and other algorithms on the benchmark functions

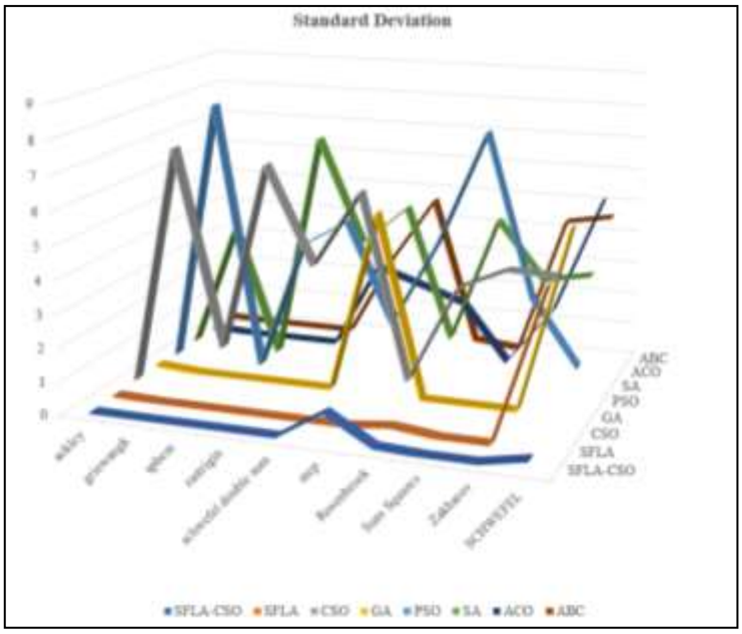

Fig. 3. STD of Hybrid SFLA-CSO, and other algorithms on the benchmark functions

Table VII shows the *R* value obtained for each algorithm than the SFLA-CSO algorithm. For example, *R*=7, means that the SFLA-CSO algorithm is seven times faster than the i^{th} algorithm in terms of the average iteration number. Accordingly, the experimental results show that the hybrid SFLA-CSO needs seven times fewer iterations to converge, than the iterations required for the standard CSO algorithm. Similarly, the hybrid SFLA-CSO needs five times less iteration to converge than the iterations required for the standard SFLA algorithm. Also, the results show that the hybrid SFLA-CSO method significantly faster than GA (9 times), PSO (16 times), SA (17 times), ACO (12 times) and ABC (17 times) methods.

Table VII. Mean of convergence speed ration based on average iteration

Algorithm	Sum of the mean iteration	Mean (Avg_Iteration)	*R*
SFLA	10373	1037.3	5.52 ≈ 5
CSO	14152	1415.2	7.52 ≈ 7
SFLA-CSO	1882	188.2	-
GA	17568	1756.8	9.33 ≈ 9
PSO	30938	3093.8	16.44 ≈ 16
SA	32273	3227.3	17.15 ≈ 17
ACO	23375	2337.5	12.42 ≈ 12
ABC	32483	3248.3	17.26≈ 17

5. Conclusion

In this research, we used metaheuristic techniques to invent a faster, better, and more efficient solution for optimisation problems. We proposed a hybrid algorithm based on the shuffled frog leaping algorithm (SFLA) and cat swarm optimization (CSO) so-called Hybrid SFLA-CSO. In this regard, several tests were carried out on different benchmark functions and the supremacy of the proposed model was investigated. The experimental results show that the hybrid proposed method needs seven times less iterations to converge in comparison to the iterations required for the standard CSO algorithm. Also, the hybrid proposed method needs five times fewer iterations to converge than the iterations required for the standard SFLA algorithm. We compared performance of the proposed hybrid algorithm against the famous relevant algorithms PSO, ACO, ABC, GA, and SA; the results were valuable and promising. Additionally, experimental results show that the Hybrid SFLA-CSO proposed method reduces the error rate as well. As a result, in the Hybrid SFLA-CSO algorithm, the particles can move faster towards a global optimum with enduring fewer error. Therefore, the results prove the characteristics of a high-performance computation such as iteration decreasing by early convergence and error reduction. As a future work, performance of the proposed hybrid SFLA-CSO on some applied problems can be investigated. Also, since the shuffled frog leaping algorithm has the potential to be parallelized, it is possible to extend it for online and even real-time systems. As a future research work, performance of the proposed algorithm against modified/improved versions of CSO or SFLA, can be investigated.

References

[1] S.Nejatian, R.Omidvar, H.Parvin, V. Rezaei, M.Yasrebi, “A new algorithm: wild mice colony algorithm (WMC)”, *Tabriz Journal of Electrical Engineering*, vol. 49, no. 1, pp. 425-437, 2019 (in Persian).

[2] A. Afroughinia and R. Kardehi Moghaddam, “Competitive learning:A new metaheuristic optimization algorithm”, *International Journal on Artificial Intelligence Tools*, vol. 27, no. 08, p. 1850035, 2018.

[3] M. Mohammadpour, H. Parvin, “Chaotic genetic algorithm based on clustering and memory for solving dynamic optimization problems”, *Tabriz Journal of Electrical Engineering*, vol.46, no.3, pp. 299-318, 2016 (in Persian).

[4] C.-W. Tsai, W.-Y. Chang, Y.-C. Wang and H. Chen, “A high-performance parallel coral reef optimization for data clustering”, *Soft Computing*, vol. 23, no. 19, pp. 1-14, 2019.

[5] X. Nie, W. Wang, H. Nie, “Chaos quantum-behaved cat swarm optimization algorithm and its application in the PV MPPT”, *Computational Intelligence and Neuroscience*, vol. 2017, 2017.

[6] Wang L, Gong Y, “A fast shuffled frog leaping algorithm”, In Ninth International Conference on Natural Computation (ICNC), July 2013, IEEE, pp. 369-373.

[7] M. M. Eusuff and K. E. Lansey, “Optimization of water distribution network design using the shuffled frog leaping algorithm”, *Journal of Water Resources planning and management*, vol. 129, no. 3, pp. 210-225, 2003.

[8] S.-C. Chu, P.-W. Tsai and J.-S. Pan, “Cat swarm optimization”, In Pacific Rim International Conference on Artificial Intelligence, August 2006, Springer, Berlin, Heidelberg, pp. 854-858.

[9] C. Jingcao, R. Zhou, and D. Lei, “Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks”, *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103540, 2020.

[10] T. Jianxin, R. Zhang, P. Wang, Z. Zhao, L. Fan, and X. Liu, “A discrete shuffled frog-leaping algorithm to identify influential nodes for influence maximization in social networks” *Knowledge-Based Systems*, vol. 187, p. 104833, 2020.

[11] A. Kaveh, S. Talatahari and N. Khodadadi, “Hybrid invasive weed optimization-shuffled frog-leaping algorithm for optimal design of truss structures”, *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, pp. 1-16, 2019.

[12] R. Dash, R. Dash and R. Rautray, “An evolutionary framework-based microarray gene selection and classification approach using binary shuffled frog leaping algorithm”, *Journal of King Saud University-Computer and Information Sciences*, 2019.

[13] R. Dash, "Performance analysis of a higher order neural network with an improved shuffled frog leaping algorithm for currency exchange rate prediction", *Applied Soft Computing*, vol. 67, pp. 215-231, 2018.
[14] T. K. Sharma and M. Pant, "Identification of noise in multi noise plant using enhanced version of shuffled frog leaping algorithm", *International Journal of System Assurance Engineering and Management*, vol. 9, no. 1, pp. 43-51, 2018.
[15] K. Daoden and T. Thaiupathump, "Applying shuffled frog leaping algorithm and bottom left fill algorithm in rectangular packing problem", In 2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), July 2017, IEEE, pp. 136–139.
[16] P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm", *Journal of Parallel and Distributed Computing*, vol. 101, pp. 41–50, 2017.
[17] D. Lei, Y. Zheng and X. Guo, "A shuffled frog leaping algorithm for flexible job shop scheduling with the consideration of energy consumption", *International Journal of Production Research*, vol. 55, no. 11, pp. 3126–3140, 2017.
[18] R. Chompu-Inwai and T. Thaiupathump, "Optimal cost driver selection in activity- based costing using shuffled frog leaping algorithm", Proceedings of the International Conference on Industrial Engineering and Operations Management, April 2017, Rabat, Morocco.
[19] Villa, Trinidad Castro, and Oscar Castillo. "Adaptation of Parameters with Binary Cat Swarm Optimization Algorithm of Controller for a Mobile Autonomous Robot", In Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine, Springer, Cham, pp. 35-46, 2020.
[20] M. Shahid Ali, A. Ahmad, J. Shafique, "Integer cat swarm optimization algorithm for multiobjective integer problems", *Soft Computing*, vol. 24, no. 3, pp. 1927-1955, 2020.
[21] R. Soto, B. Crawford, A. Aste Toledo, C. Castro, F. Paredes, R. Olivares et al., "Solving the manufacturing cell design problem through binary cat swarm optimization with dynamic mixture ratios", *Computational Intelligence and Neuroscience*, vol. 2019, 2019.
[22] L.Pappula, D. Ghosh, "Cat swarm optimization with normal mutation for fast convergence of multimodal functions", *Applied Soft Computing*, vol. 66, pp. 473–491, 2018.
[23] M. Zhao, "A novel compact cat swarm optimization based on differential method", *Enterprise Information Systems*, vol. 14, no. 2, pp. 196-220, 2020.
[24] A. Thomas, P. Majumdar, T. Eldho, A. Rastogi, "Simulation optimization model for aquifer parameter estimation using coupled meshfree point collocation method and cat swarm optimization", *Engineering Analysis with Boundary Elements*, vol. 91, pp. 60–72, 2018.
[25] V. I. Skoullis, I. X. Tassopoulos, G. N. Beligiannis, "Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm", *Applied Soft Computing*, vol. 52, pp. 277–289, 2017.
[26] H. Chen, Q. Feng, X. Zhang, S. Wang, W. Zhou, Y. Geng, "Well placement optimization using an analytical formula-based objective function and cat swarm optimization algorithm", *Journal of Petroleum Science and Engineering*, vol. 157, pp. 1067–1083, 2017.
[27] B. Kumar, M. Kalra, P. Singh, "Discrete binary cat swarm optimization for scheduling workflow applications in cloud systems", In 2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT), February 2017, IEEE, pp.1–6.
[28] D. Diana, "Novel cat swarm optimization algorithm to enhance channel equalization", *COMPEL- The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 36, no. 1, pp. 350–363, 2017.
[29] S.-H. Wang, W. Yang, Z. Dong, P. Phillips, Y.-D. Zhang, "Facial emotion recognition via discrete wavelet transform, principal component analysis, and cat swarm optimization", In International Conference on Intelligent Science and Big Data Engineering, September 2017, Springer, Cham, pp. 203–214.
[30] E. N. Kencana, N. Kiswanti, K. Sari, "The application of cat swarm optimization algorithm in classifying small loan performance", Journal of Physics: Conference Series, vol. 893, no. 1, IOP Publishing, p. 012037, 2017.
[31] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, A. Al-Khasawneh, "Cloud scalable multi-objective task scheduling algorithm for cloud computing using cat swarm optimization and simulated annealing", In 2017 8th International Conference on Information Technology (ICIT), May 2017, IEEE, pp. 599–604.
[32] K. K. Dhaliwal, J. S. Dhillon, "Integrated cat swarm optimization and differential evolution algorithm for optimal IIR filter design in multi-objective framework", *Circuits, Systems, and Signal Processing*, vol. 36, no. 1, pp. 270–296, 2017.
[33] M. Jamil, X.-S. Yang, "A literature survey of benchmark functions for global optimization problems", *arXiv preprint* arXiv:1308.4008, 2013.