Applied Computing and Intelligence

https://www.aimspress.com/journal/aci

Applied Computing and Intelligence
1(1): 75–94
DOI: 10.3934/aci.2021005
Received: 09 December 2021
Accepted: 27 December 2021
Published: 30 December 2021

Research article

Adjustable mode ratio and focus boost search strategy for cat swarm optimization

Pei-Wei Tsai[1,*], **Xingsi Xue**[2,3], **Jing Zhang**[2,4] **and Vaci Istanda**[5]

[1] Swinburne University of Technology, Australia
[2] Fujian Provincial Key Laboratory of Big Data Mining and Applications, FJUT, China
[3] Center for Information Development and Management, FJUT, China
[4] Fujian University of Technology (FJUT), China
[5] Indigenous Peoples Commission, Taipei City Government, Taiwan (R.O.C.)
* **Correspondence:** ptsai@swin.edu.au

Academic Editor: Chih-Cheng Hung

Abstract: Evolutionary algorithm is one of the optimization techniques. Cat swarm optimization (CSO)-based algorithm is frequently used in many applications for solving challenging optimization problems. In this paper, the tracing mode in CSO is modified to reduce the number of user-defined parameters and weaken the sensitivity to the parameter values. In addition, a *mode ratio* control scheme for switching individuals between different movement modes and a search boosting strategy are proposed. The obtained results from our method are compared with the modified CSO without the proposed strategy, the original CSO, the particle swarm optimization (PSO) and differential evolution (DE) with three commonly-used DE search schemes. Six test functions from IEEE congress on evolutionary competition (CEC) are used to evaluate the proposed methods. The overall performance is evaluated by the average ranking over all test results. The ranking result indicates that our proposed method outperforms the other methods compared.

Keywords: evolutionary algorithm; cat swarm optimization; dynamic parameter control; swarm intelligence; particle swarm optimization; differential evolution

1. Introduction

Applying optimization in practice usually requires input from domain experts to provide insights and experiences for aiding the optimization process to be more efficient. The domain knowledge is beneficial for setting up boundaries and criteria for sieving out feasible solutions. Nevertheless,

converting the domain knowledge to parameter settings for an evolutionary algorithm (EA) is not intuitive and is challenging when more parameters require tuning.

Current solutions for different EAs rely on the domain experts to be involved in the optimization process for designing the proper parameter values in the real-world applications. Tuning more parameters is more costly in terms of time consuming and workload. Thus, reducing the number of parameters for tuning is an important subject of study.

The key idea of how we plan to improve the process is to reduce the number of user-defined parameters for tuning. Among existing EAs, we select CSO as the target algorithm for improving its searching ability in this work and compare the results with other famous EAs because CSO has a unique feature that enables the flexibility to enhance either the exploration or the exploitation ability.

Cat swarm optimization (CSO) (proposed by Chu et al. in 2006 [1]) is an EA for optimization built based on mimicking the behavior of cats. It has two modes corresponding to different movement behaviors which dominate the exploration and exploitation in solution finding. The exploration ability corresponds to the global search ability while the exploitation can be mapped to the local search capacity. CSO has been applied to multiple real-world applications such as creating text summarization [2], predicting trends or data [3], solving backboard wiring problems [4], solving scheduling problems [5, 6], balancing load on cloud computing [7], identifying overlapping communities in social networks [8], allocating components in distribution systems [9], deploying nodes in wireless sensor network [10], tracking the multipeak maximum power point for photo-voltaic array under complex conditions in solar power system [11], classifying data [12], selecting features [13, 14], grouping data by clustering [15], partitioning data [16, 17], recognizing facial emotions [18, 19], suppressing noise [20], estimating motions in video sequence block matching [21], hiding information via video steganography [22], solving routing problems for vehicles [23], saving transmission power [24], and many other fields. According to the survey works [25, 26] on CSO related algorithms and applications, it outperforms other EAs in many cases and shows the potential for solving NP-hard problems.

In any EA, parameter tuning is always a vital task because the user-defined parameter values significantly impact the performance of the algorithm and the quality of the obtained solutions. Integrating knowledge from experts in the application field is highly recommended when it comes to a practical application. Nevertheless, EAs are built for general purposes. Thus, in most cases, the parameters are not directly linked to the domain knowledge but are used to control the search behavior of the algorithm. This increases the difficulty for the general public, who has no experience in EAs, to use them in problem-solving.

To overcome this issue, the solution can be roughly categorized in two ways: reducing the number of user-defined parameters or conducting a comprehensive analysis to understand the terrain of the solution space before deciding the parameter values. Since the analysis is time-consuming and is not always practical, the former potential solution is more reasonable. However, the user-defined parameters in EAs are designed for shaping the search behavior of the algorithm. Simply removing the parameters from the equation or replacing them with constants may cause the degradation in the algorithm performance. To compensate the potential drop in algorithm performance, corresponding designs and strategies such as the *mode ratio* control scheme and the search boosting strategy can be enhanced to assist the EA in solution-finding.

The other important factors affecting the quality of the obtained solution in EAs are the diversity of the individuals in the population and the local exploitation ability. A more flexible learning or updating

strategy enables the EA to maintain higher diversity for more substantial exploration ability [27]. In contrast, the delicate exploitation strategy improves the local search capacity to fine-tune the obtained solution. In this work, we propose a dynamic *mode ratio* control strategy for navigating the individuals to focus on the exploration and gradually increasing the exploitation. Moreover, a focus boost search strategy is introduced to narrow down the search scope further and allocate all available resources to the most potential solution before consuming all available resources.

The remaining article is organized as follows: the review on CSO is given in Section 2, the proposed modification and strategies are described in Section 3. The experiments and experimental results are revealed in Section 4 and followed by the conclusion and future work.

2. Review on cat swarm optimization

Many optimization problems can be classified in the NP-hard problems [28]. For the NP-hard problems, conventional mathematical solvers are not applicable because the computational complexity is too high and thus it won't obtain a solution within a reasonable time. Methods including a meta-heuristic component such as EA are more suitable for solving NP-hard problems. Examples of optimization in the NP-hard level include optimizing the deployment locations for maximize the power efficiency in the Wireless Sensor Networks (WSN) [29] and optimizing the ontology matching results [30].

Many different modifications have been proposed by researches in the past decades. The parallel CSO (PCSO) (proposed by Tsai et al. in 2008 [31]) divides the swarm into subswarms and then eliminates the worst solution in each subswarm. A parameter *ECH* is introduced in this work to control the information exchange between subswarms. PCSO can be used in the resource limited environment with small population.

Santosa and Ningrum first apply CSO in the clustering problems for finding the best cluster centers in 2009 [32]. They remove the parameter *MR* to force all individuals be updated by both seeking and tracing modes in every generation. Moreover, they suggest to pull up the value of the parameter *CDC* to 1 to ensure all dimensions are updated in every generation to increase the diversity.

Average-inertia weighted CSO (AICSO) (proposed by Orouskhani et al. in 2011 [36]) introduces a weight parameter ω for adjusting the importance of the existing velocity in solution update. Moreover, the solution update formula for the tracing mode is changed to the sum of both the mean position and velocity between the previous and the current generations.

The multi-objective CSO (MOCSO) (proposed by Pradhan and Panda in 2012 [35]) utilizes an external archive and the Pareto dominance to handle the nondominated solutions. The design aims at enabling MOCSO to work on the multi-objective optimizations.

The enhanced parallel CSO (EPCSO) (proposed by Tsai et al. in 2012 [5]) adopts the orthogonal array from the Taguchi method into the tracing mode of PCSO. Two sets of candidate velocities are generated and sampled by Taguchi method to form the candidate solutions. After evaluating the candidates' fitness value, the selection process creates the update information and indicates the corresponding sources.

The discrete binary CSO (BCSO) (proposed by Sharafi et al. in 2013 [33]) uses a *probability of mutation operation* (*PMO*) parameter to replace *SRD* in the seeking mode, and the tracing mode is modified to create two binary velocity vectors by taking contents from either the current position

vector or the global position vector.

The adaptive dynamic CSO (ADCSO) (proposed by Orouskhani et al. in 2013 [34]) introduces an adjustable inertia to the velocity update formula in the tracing mode. It is suggested that the inertia should be decreased while the problem dimension increases. The constant C in the tracing mode velocity update formula is also substituted by an adjustable value. Moreover, the average of the neighborhood dimensions is used in part of the solution update for bringing in higher diversity.

The improvement structure of CSO (ICSO) (proposed by Hadi and Sabah in 2015 [21]) combines the information exchange from PCSO and the inertia weight for the tracing mode from AICSO and applies the ICSO in motion estimation in block matching to avoid the degradation of the image quality.

Chaos quantum-behaved CSO (CQCSO) (proposed by Nie et al. in 2017 [11]) combines the CSO with quantum mechanics and incorporates a tent map technique for avoiding trapping in the local optima. It is used in the photovoltaic maximum power point tracking model control and verified that CQCSO produces the global maximum power point racking control strategy in their experiments.

In short, most of the existing modifications focus on hybrid structures for adopting concepts from other methods to increase the solution diversity and the search ability. They are partially customized for the selected applications. We look at the common problems on those modified CSOs and directly improve the original CSO, instead of using the hybrid structure approach used by most other modified CSOs. Hence, our proposed method is different in this aspect, and it reduces the computational complexity than before the modification. The computational cost increases along with the increment of the computational complexity. To avoid further increasing the complexity, we choose the original CSO as the start point. The original CSO is one of the existing EAs with a controllable parameter called *mixture ratio* (*MR*) [1] for deciding the movement behavior of the individuals in the solution update process. The origin of designing this mechanism is to allow users to adjust the search ability for adopting different terrains of the objective functions. Nevertheless, this information is not available without several tryouts for an average end-user or can only be predicted by experts already familiar with the given objective function in the field. Aside from *MR*, four parameters called *SMP*, *SPC*, *CDC*, and *SRD* and a constant C are used in its seeking mode and the tracing mode, respectively [1]. In this work, we rename the *SMP*, *SPC*, *CDC*, and *SRD* to α, β, γ, and δ, respectively, for simplification. The definition and the corresponding symbols used in this work are summarized below:

- *SMP* - This parameter stands for the *seeking memory pool* [1], which manages the number of candidate solutions involved in the seeking mode process. In this work, α is used to represent *SMP*.
- *SPC* - This parameter is presented by β in this work. It came from *self position consideration* [1] for indicating whether the current solution is included in the candidate solutions.
- *CDC* - *CDC* indicates the *counts of dimension to change* [1], which contains the proportion of dimensions that will be modified for the candidate solutions. γ is used to present *CDC* in this work.
- *SRD* - It means the *seeking range of the selected dimension* [1]. This parameter defines the variable range on a selected dimension for the candidate solutions. Symbol δ is used to present *SRD* in this work.

In the following sections, we use symbols $\mathbf{X}$, $\mathbf{V}$, and $\mathbf{m}$ to represent the solution, the velocity, and the movement mode of the individuals in the population, respectively. Boundaries for $\mathbf{X}$ and $\mathbf{V}$ are

defined by the users depending on the feasible range of the solutions and the complexity of the solution space. Usually, these boundaries very with different scenarios.

2.1. Problem definition

Let the total number of individuals in the population denoted by n, CSO evolves individuals in d-dimensional, which can be described as:

$$\mathbf{X}_g = \left\{\mathbf{x}_{1,g}, \cdots, \mathbf{x}_{n,g}\right\} \tag{1}$$

and

$$\mathbf{x}_{i,g} = \left\{x_{i,g}^1, \cdots, x_{i,g}^d\right\} \tag{2}$$

where g stands for the g^{th} generation, $\mathbf{x}_{i,g}$ is the i^{th} individual, and $\mathbf{X}_g$ represents the set of all individuals at the g^{th} generation.

The j^{th} element $x_{i,0}^j$ in the initialization at $g = 0$ of the i^{th} target vector is assigned values by select random value between the boundaries defined by min and max:

$$x_{i,0}^j = x_{min}^j + r \times (x_{max}^j - x_{min}^j) \tag{3}$$

where r is a uniformly distributed random number in $[0, 1]$, x_{min}^j denotes the lower bound value at dimension j, and x_{max}^j is the upper bound value of the j^{th} dimension.

Corresponding to each individual, a velocity vector and the movement mode indicator vector are also initialized:

$$\mathbf{V}_g = \left\{\mathbf{v}_{1,g}, \cdots, \mathbf{v}_{n,g}\right\} \tag{4}$$

$$\mathbf{v}_{i,g} = \left\{v_{i,g}^1, \cdots, v_{i,g}^d\right\},\ v_{min}^j \leq v_{i,g}^j \leq v_{max}^j,\ j \in [1, d] \tag{5}$$

and

$$\mathbf{m}_g = \left\{m_{1,g}, \cdots, m_{n,g}\right\} \tag{6}$$

where $\mathbf{v}_{i,g}$ is the velocity belongs to the i^{th} individual at the g^{th} generation, v_{min}^j and v_{max}^j are the lower and upper bounds of the velocity on the j^{th} dimension, $m_{i,g}$ is a binary flag indicating the movement mode of the i^{th} individual at generation g, and the proportion of $\mathbf{m}_g$ is controlled by MR [1].

The velocity bounds are only used in the original CSO [1]. They are removed in our proposed method because the momentum is eliminated in our proposed method.

2.2. Evaluation

The objective function is a user defined function for measuring whether a solution is better than another. The objective function must reflect the problem to be solved. Assume that the goal is to minimize the function value, after initializing all target vectors, $\mathbf{X}_g$ is evaluated by the objective function, and the fitness values are kept in $\mathbf{f}_g$:

$$\mathbf{f}_g = \left\{ f_{1,g}, \cdots, f_{n,g} \right\} \tag{7}$$

$$f_{i,g} = Eval(\mathbf{x}_{i,g}), \forall i \tag{8}$$

and

$$f_{best,g} = min(min(\mathbf{f}_g), f_{best,g-1}) \tag{9}$$

where $f_{i,g}$ is the fitness value of the i^{th} individual at the g^{th} generation, $Eval(\cdot)$ stands for the objective function, $f_{best,g}$ and $f_{best,g-1}$ represent the best fitness values obtained at the current generation and at the previous generation.

2.3. Solution update

After updating $f_{best,g}$, CSO updates the individual i with either seeking mode or tracing mode by referring to $m_{i,g}$. For individuals in the tracing mode, Eqs (10)–(12) are used to update $\mathbf{x}_{i,g}$ in Eq (2):

$$vcad^j_{i,g} = v^j_{i,g-1} + r \times C \times (x^j_{best,g} - x^j_{i,g-1}) \tag{10}$$

$$v^j_{i,g} = \begin{cases} v^j_{max}, & \text{if } vcad^j_{i,g} > v^j_{max} \\ v^j_{min}, & \text{if } vcad^j_{i,g} < v^j_{min} \\ vcad^j_{i,g}, & \text{otherwise} \end{cases} \tag{11}$$

and

$$x^j_{i,g} = x^j_{i,g-1} + v^j_{i,g} \tag{12}$$

where $vcad^j_{i,g}$ is the candidate velocity of individual i at the g^{th} generation on the j^{th} dimension, C is a constant, and $x_{best,g}$ indicates the current best solution.

On the other hand, if the $m_{i,g}$ brings the i^{th} individual into the seeking mode, α copies of $\mathbf{x}_{i,g}$ are generated in the candidate vector:

$$\mathbf{X}_{i,g,cad} = \left\{ \mathbf{x}_{i,g,k_0}, \cdots, \mathbf{x}_{i,g,\alpha_0} \right\} \tag{13}$$

where $\mathbf{x}_{i,g,k_0}$ indicates the k^{th} copy of $\mathbf{x}_{i,g}$ and the sub-index 0 is used to represent the copies before the modification.

The reason to create α copies and make slight modifications on the copies is to create solutions that are physically nearby the original solution in the solution space. This process helps to increase the local exploitation ability of the algorithm [1].

A Boolean variable (β) is used for indicating whether to keep an unmodified copy in the candidate. Assume β indicates keeping an unmodified copy, only $\alpha - 1$ copies will be modified. For the candidate dimensions to be modified, randomly select γ percent of the dimensions on $\mathbf{x}_{i,g,k_0}$ and update the corresponding dimensions by:

$$x^{j}_{i,g,k_1} = \begin{cases} (1 + r_1 \times \delta) \times x^{j}_{i,g,k_0}, & \text{if } j \text{ is the selected dimension to be updated} \\ x^{j}_{i,g,k_0}, & \text{otherwise} \end{cases} \tag{14}$$

where $r_1 \in [-1, 1]$ is a random variable, $\delta \in [0, 1]$ is a proportion for updating x^{j}_{i,g,k_1} and the sub-index 1 is used to indicate the candidates after the values on some dimensions are modified.

The fitness value of all candidates is calculated by Eq (9) and stored in a vector:

$$\mathbf{f}_{i,g,cad} = \left\{ f_{i,g,k_1}, \cdots, f_{i,g,\alpha_1} \right\} \tag{15}$$

where $\mathbf{f}_{i,g,cad}$ is the vector containing fitness values corresponding to the candidates.

A vector indicating the probability for selecting a particular candidate ($\hat{\mathbf{P}}_i$) is calculated, and then one of the candidates is selected to replace individual i by roulette wheel selection. It is noticeable that Eq (18) has excluded the candidate, which has the worst fitness value, in the selection process. This operation involves a hybrid concept of elite selection and heuristic learning.

$$\hat{\mathbf{P}}_i = Normalization(\mathbf{P}_i) \tag{16}$$

where $Normalization(\cdot)$ is a min-max normalization function that converts the input into the range in $[0, 1]$.

$$\mathbf{P}_i = \left\{ p^{k}_{i}, \cdots, p^{\alpha}_{i} \right\} \tag{17}$$

$$p^{k}_{i} = \frac{|f_{i,g,k_1} - f_{i,g,\zeta}| + \epsilon}{max(\mathbf{f}_{i,g,cad}) - min(\mathbf{f}_{i,g,cad}) + \epsilon} \tag{18}$$

and

$$f_{i,g,\zeta} = \begin{cases} max(\mathbf{f}_{i,g,cad}), & \text{if the goal is to minimize the function value} \\ min(\mathbf{f}_{i,g,cad}), & \text{otherwise} \end{cases} \tag{19}$$

where p^{k}_{i} indicates the probability of selecting the k^{th} candidate for replacing the i^{th} individual and ϵ is a tinny positive constant to avoid Eq (18) to be divided by zero. Eq (19) is used for determining $f_{i,g,\zeta}$ according to the optimization goal.

Assume that the goal of optimization is to find the solution to minimize the fitness value, the candidate with the worst solution would have the maximum fitness value comparing to other candidates. Thus, the numerator in Eq (18) is equal to ϵ, thus the probability for the worst candidate is approximated to zero. To the contrary, the candidate with the best solution would get $p^{k}_{i} = 1$ because the numerator will be identical to the denominator in Eq (18). Hence, the probability of selecting the candidate with the best fitness value is greater than others after normalizing the probability in Eq (16).

After updating all the individuals by either way mentioned above, $\mathbf{m}_g$ in Eq (6) is reinitialized (see Algorithm 4) for selecting individuals to be updated by tracing mode in the next generation. The process goes back and forward between the evaluation and the solution update until the termination criteria are satisfied for terminating the process and output the obtained near best solution.

3. Proposed strategies

Table 1 lists all parameters and their functionalities used in the proposed modification of CSO.

Table 1. Parameters in the modified CSO.

Parameter	Data Type/Range	Functionality
α	*Integer*	Manages the number of candidate solutions to be generated in the seeking mode.
β	*Boolean*	Determines whether $\mathbf{x}_{i,g}$ should be included in $\mathbf{X}_{i,g,cad}$.
γ	[0, 1]	Defines the proportion of dimensions in a candidate solution to be updated.
δ	[0, 1]	Defines the variable proportion of values for the selected $x^j_{i,g,k}$.
τ	[0, 1]	Indicates the proportion of the swarm be updated in the tracing mode (Algorithm 1).
λ	[0, 1]	Indicates the preserved proportion of function evaluation values (FEVs) for concentrating the resource on the most promising solution in the swarm.

By looking at the processes in CSO, we know that many user-defined parameters are included in the solution-finding process, and each of the parameters can potentially affect the solution quality. However, we also notice that some of the processes can be further simplified by implementing corresponding strategies. The upper and the lower bounds of the velocity are two parameters that are challenging for the users to decide because there is no significant reference information to guide the user on how to make the proper setting in their applications. The quality of the obtained solution can be significantly different only by using different values for the velocity bounds (see Fig. 1).

Fig. 1 reveals the results of Ackley test function [37] from CEC competition [38] with identical parameter settings except for the velocity bounds in CSO. The x-axis indicates the Function Evaluation Values (FEV), which shows the count of function evaluations, and the y-axis is the fitness value from the objective function. This example shows the potential issue caused by an improper parameter setting. When the velocity bounds are too large, CSO has difficulty in converging to the right spot. PSO also has the same issue that the velocity bounds can significantly affect the result. A higher velocity bound gives a higher exploration quota to the individual but would flatten the exploitation ability at some points. Furthermore, the purpose of Eqs (10) and (11) mainly introduces the distance between the known best solution to the current individual. Taking the concept from Differential Evolution [39], a replacement for Eqs (10)–(12) is proposed:

$$x^j_{i,g} = x^j_{i,g-1} + r \times C \times (x^j_{best,g} - x^j_{i,g-1}) \tag{20}$$

In Eq (20), the distance between the current best solution to the i^{th} individual is used directly to guide the individual in the tracing mode. This strategy removes the velocity bounds from the equation and thus simplifies the structure of the whole algorithm. By eliminating the velocity from the $g - 1$ generation, the search is fully guided by the differences at present but not influenced by the inertia accumulated from the past.

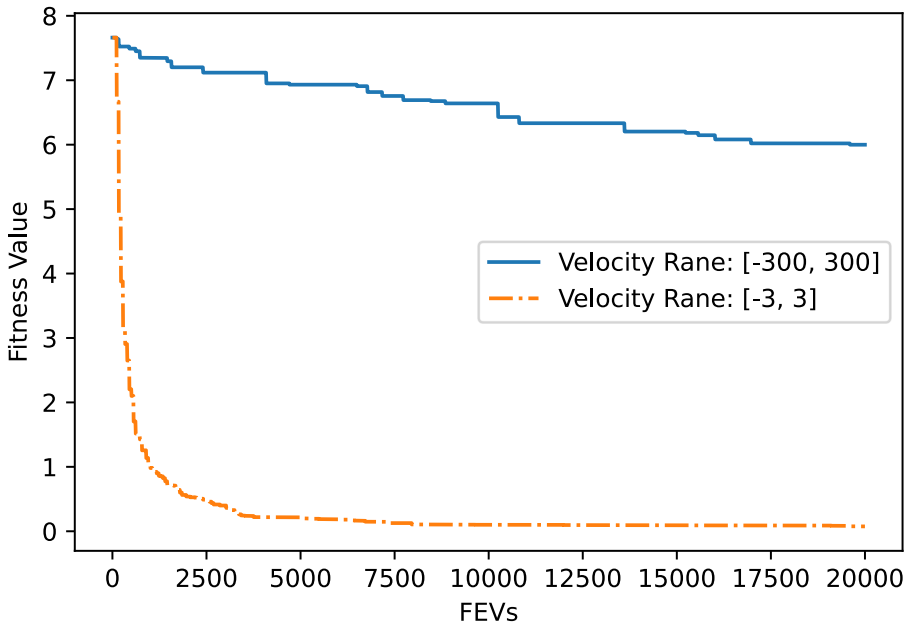

Figure 1. Example fitness values obtained with Ackley test function by CSO.

A simple example for comparing the differences between the tracing mode from the original CSO and the modified tracing mode is given as follows. Assume that $x^j_{best,g}$, $x^j_{i,g-1}$, r, C, $v^j_{i,g-1}$, v^j_{max}, and v^j_{min} equal to 5, −2, 0.7, 2, 2, 3, and −3, respectively. After updating solution i on the j^{th} dimension by the tracing mode, the obtained $x^j_{i,g}$ will be 1 in the original CSO [1] and 7.8 in the proposed method, respectively.

On the other hand, a variable *mode ratio* (τ) is proposed to replace the fixed MR in CSO to compensate for the reduced exploration ability by replacing Eqs (10)–(12) with Eq (20). In the original CSO, the tracing mode is used to enrich the exploration ability, while the seeking mode is designed to provide the algorithm's exploitation capacity. Thus, τ is recommended to shift from a larger value toward a smaller value for directing more individuals into the tracing mode at the beginning and gradually allocate more individuals into the seeking mode toward the end of the search. In practice, γ and δ are utilized as the bounds of τ.

Moreover, a focus boost search strategy is introduced to the process and triggered after the remaining FEV budget is within a certain proportion (λ). The FEV is the maximum count for calling the objective function to evaluate the solution. In general, the individuals should have a sufficient quota for exploring the solution space in the early stage of the search. When the search process is approaching the end, the focus boost search strategy forces CSO to move the only individual, which holds the best fitness value at the current generation, in seeking mode till the termination of the search process. This strategy increases the exploitation capacity of CSO to focus on discovering potential solutions near the current best solution and avoiding wasting FEVs on exploring new regions while the search process is close to the end.

The pseudo-code of the modified tracing mode and seeking mode operations are given in Algorithm 1 and Algorithm 2, respectively, while the pseudo-code for the CSO with the proposed changes is

provided in Algorithm 3. The E_{max} in the input of Algorithm 3 is the maximum allowed FEVs used as the termination criterion for the search. Algorithm 4 is used for updating the $\mathbf{m}_g$ vector.

Algorithm 1: Modified tracing mode

input : $\{\mathbf{x}_{i,g-1}, \mathbf{x}_{best,g}, C, d\}$
output: $\mathbf{x}_{i,g}$

for $j \leftarrow 1$ **to** d **by** 1 **do**
  $x^j_{i,g} \leftarrow x^j_{i,g-1} + r \times C \times (x^j_{best,g} - x^j_{i,g-1})$
end

4. Experiments

4.1. Test problems

To test the search ability of CSO with our proposed strategies and modification, six benchmark functions, including Ackley, Rastrigin, Griewank, Sphere, Rosenbrock, and Weierstrass, from the CEC competition are used in the experiments. These test functions from the CEC competition are designed for evaluating the accuracy of algorithms on finding global optimum solutions. Each test function contains lots of local optimum, which are potentially traps to the algorithms. If an algorithm is not capable of jumping out from the local optimum, the accuracy of finding the global optimum solution would be low. On the other hand, if an algorithm escapes from the local optimum traps, the chance for it to find the global optimum solution is relatively high. An algorithm is treated as with high accuracy if the final solution it produces is very close to the global optimum value.

4.2. Test setup

The dimension of the test function is set to $d = 50$ for each function, except Ackley is typically designed as a 2-dimensional function. The optimization tasks are carried out by the original CSO [1], particle swarm optimization (PSO) [42], DE with three commonly-used search schemes [43] ("DE/rand/1/bin", "DE/best/1/bin", and "DE/current-to-best/1/bin"), and CSO with our proposed strategies and modification. A total of $20,000$ FEVs is given to every method as the budget for searching solutions for each test function.

The parameters in PSO and the original CSO are set by the recommended values in [40] while the velocity bounds are set to $[-3, 3]$ and MR is set to 0.05 for CSO in all functions because not all test functions are used in the literature, and the dimension used in the test functions are also higher to make the task more challenging. The crossover rate (c_r) and the scaling factor (F) for DE are set to 0.9 and 0.5, respectively, as recommended in [41].

CSO with our proposed strategies and modification follows the same setting used in [40] except the newly introduced τ is allowed to linearly decreases from γ to δ, and λ is set to the same value as δ. The number of individuals used in all methods is set to $n = 50$. All test functions' initialization range is bounded in $[-30, 30]$ on each dimension. All experiments are performed with 30 independent runs. The mean of the fitness values obtained from all independent runs is reported as the final result to measure the performance of an algorithm in terms of optimization accuracy.

Algorithm 2: Seeking mode

input : $\{\mathbf{x}_{i,g-1}, \mathbf{f}_{i,g-1}, d, \alpha, \beta, \gamma, \delta\}$
output: $\mathbf{x}_{i,g}$

$\mathbf{X}_{i,g,cad} \leftarrow \emptyset$
$\mathbf{f}_{i,g,cad} \leftarrow \emptyset$
$\mathbf{P}_i \leftarrow \emptyset$
for $k \leftarrow 1$ **to** α **by** 1 **do**
    $\mathbf{x}_{i,g,k} \leftarrow \mathbf{x}_{i,g}$
end
if β **then**
    $s \leftarrow 2$
    $\mathbf{X}_{i,g,cad} \leftarrow \mathbf{x}_{i,g,1}$
    $f_{i,g,1} \leftarrow \mathbf{f}_{i,g-1}$
    $\mathbf{f}_{i,g,cad} \leftarrow f_{i,g,1}$
else
    $s \leftarrow 1$
end
for $k \leftarrow s$ **to** α **by** 1 **do**
    $\mathbf{t} \leftarrow R(1, j)$, where $R(1, j)$ is a random permutation sequence containing values in [1, j]
    for $j \leftarrow 1$ **to** $(\gamma \times d)$ **by** 1 **do**
        $x_{i,g,k_1}^{\mathbf{t}_j} \leftarrow (1 + r_1 \times \delta) \times x_{i,g,k_0}^{\mathbf{t}_j}$, where $\mathbf{t}_j$ stands for the j^{th} element in $\mathbf{t}$
    end
    for $j \leftarrow [(\gamma \times d) + 1]$ **to** d **by** 1 **do**
        $x_{i,g,k_1}^{\mathbf{t}_j} \leftarrow x_{i,g,k_0}^{\mathbf{t}_j}$
    end
    $\mathbf{X}_{i,g,cad} \leftarrow \mathbf{x}_{i,g,k}$
    $f_{i,g,k} \leftarrow Eval(\mathbf{x}_{i,g,k})$
    $\mathbf{f}_{i,g,cad} \leftarrow f_{i,g,k}$
end
for $k \leftarrow 1$ **to** α **by** 1 **do**
    $p_i^k \leftarrow \frac{|f_{i,g,k_1} - f_{i,g,\zeta}| + \epsilon}{max(\mathbf{f}_{i,g,cad}) - min(\mathbf{f}_{i,g,cad}) + \epsilon}$
    $\mathbf{P}_i \leftarrow p_i^k$
end
$\hat{\mathbf{P}}_i \leftarrow Normalization(\mathbf{P}_i)$
$\mathbf{x}_{i,g} \leftarrow Roulette(\hat{\mathbf{P}}_i)$, where $Roulette(\cdot)$ means the roulette wheel selection

Algorithm 3: Modified CSO

input : $\{n, E_{max}, C, d, \alpha, \beta, \gamma, \delta, \lambda\}$
output: $\mathbf{x}_{best,g}$

$g \leftarrow 0$
$E \leftarrow 0$, where E is the count of function evaluation
$\mathbf{t} \leftarrow R(1,n)$, where $R(1,n)$ is a random permutation sequence containing values in [1, n]
$\mathbf{m}_g \leftarrow$ **Algorithm 4**$(\gamma, \delta, E, E_{max})$
$Init(\mathbf{X}_g)$, where $Init(\cdot)$ means the initialization process.
while $E \leq E_{max}$ **do**
    if $E \leq (\lambda \times E_{max})$ **then**
        for $i \leftarrow 1$ **to** n **by** 1 **do**
            $f_{i,g} \leftarrow Eval(\mathbf{x}_{i,g})$,
            $E \leftarrow E + 1$
        end
        for $i \leftarrow 1$ **to** n **by** 1 **do**
            if $m_{i,g}$ **then**
                $\mathbf{x}_{i,g+1} \leftarrow$ **Algorithm 1**$(\mathbf{x}_{i,g}, \mathbf{x}_{best,g}, C, d)$
            else
                $\mathbf{x}_{i,g+1} \leftarrow$ **Algorithm 2**$(\mathbf{x}_{i,g}, \mathbf{f}_{i,g}, d, \alpha, \beta, \gamma, \delta)$
                if β **then**
                    $E \leftarrow E + (\alpha - 1)$
                else
                    $E \leftarrow E + \alpha$
                end
            end
        end
        $\mathbf{m}_{g+1} \leftarrow$ **Algorithm 4**$(\gamma, \delta, E, E_{max})$
    else
        $\mathbf{x}_{best,g+1} \leftarrow$ **Algorithm 2**$(\mathbf{x}_{best,g}, \mathbf{f}_{best,g}, d, \alpha, \beta, \gamma, \delta)$
        if β **then**
            $E \leftarrow E + (\alpha - 1)$
        else
            $E \leftarrow E + \alpha$
        end
    end
    $g \leftarrow g + 1$
end

Algorithm 4: *Mode ratio* update

input : $\{\gamma, \delta, E, E_{max}\}$
output: $\mathbf{m}_g$

$\tau \leftarrow [\frac{E_{max}-E}{E_{max}} \times (\delta - \gamma)] + \gamma$
$\mathbf{t} \leftarrow R(1, n)$
for $i \leftarrow 1$ **to** $(\tau \times n)$ **by** 1 **do**
  $m_{\mathbf{t}_{i,g}} \leftarrow$ True
end
for $i \leftarrow [(\tau \times n) + 1]$ **to** n **by** 1 **do**
  $m_{\mathbf{t}_{i,g}} \leftarrow$ False
end
$\mathbf{m}_g \leftarrow \left\{m_{i,g}, \cdots, m_{n,g}\right\}$

4.3. Results

The mean fitness value obtained by each method is displayed in Table 2, where the DE/$c2b$/1/bin stands for "DE/current-to-best/1/bin", CSO-C indicating the original CSO algorithm, CSO-M is the CSO with the modified tracing mode, and CSO-M/τ/λ represents the CSO with the modified tracing mode plus both the variable *mode ratio* and the focus boost search strategy.

The test result from all methods are ranked by the final obtained fitness value in ascending order. For example, assume that method A outperforms the other methods by obtaining the minimum mean fitness value over 30 independent runs in a test function, method A is ranked 1 for this test. The remaining methods are ranked in sequence according to their mean fitness values. After ranking all methods for all test functions, the mean ranking across all test functions for every method is obtained as the final result. The ranked result is shown in Table 3. A method, which has the top rank, indicates that it outperforms the others more often in all experiments.

When more than one method show the same result in an experiment, they are treated equally on the same rank and the remaining methods are ranked following the same sequence as there is no equal results. Thus, all methods are ranked 1 in the Ackley function test results because all of them obtain the global optimum solution. For test functions Rastrigin, Rosenbrock, and Weierstrass, there is no equally ranked method and thus the rank is evaluated straight forward. In the Sphere function test, CSO-M and CSO-M/τ/λ obtain the same fitness value in their output. Thus, these two methods are both ranked 1 while DE/$rand$/1/bin, PSO, CSO-C, DE/$c2b$/1/bin, and DE/$best$/1/bin are ranked sequentially from 3 to 7. Since we have two methods gaining the first rank, the second rank is not used but the next rank starts from the third one. The same concept is applied to the Griewank function test result. CSO-M and CSO-M/τ/λ are ranked 1, PSO and DE/$rand$/1/bin are ranked 3, DE/$c2b$/1/bin and CSO-C are ranked 5, respectively, and DE/$best$/1/bin is ranked 7. The ranks of an algorithm is averaged over all experiments and the results are revealed in the Average Rank in Table 3 while the Final Rank column reveals the final ranking result for concluding the experiments.

In Table 3, we can see that our proposed modification to the tracing mode operation, the variable *mode ratio*, and the focus boost strategy helps CSO to produce more accurate search results. By observing Table 2, we can find that the top two ranked test results in all test functions are close to each

other except the Rosenbrock function. This means that the proposed strategies help, but the significant improvement comes from the modification of tracing mode process.

Table 2. Experimental results obtained by different methods.

	Ackley	Rastrigin	Griewank	Sphere	Rosenbrock	Weierstrass
PSO [42]	**0**	**533**	0.9	93	31,885	−0.488
DE/*best*/1/*bin* [43]	**0**	1,612	1.3	1,159	165,755	−0.496
DE/*c2b*/1/*bin* [43]	**0**	935	1.1	433	62,692	−0.482
DE/*rand*/1/*bin* [43]	**0**	560	0.9	79	27,176	−0.483
CSO-C [1]	**0**	752	1.1	226	40,899	−0.467
CSO-M (proposed)	**0**	924	**0**	**0**	1,538	**−0.514**
CSO-M/τ/λ (proposed)	**0**	802	**0**	**0**	**511**	−0.502

Table 3. Average ranking results.

	Average Rank
PSO [42]	2.33
DE/*best*/1/*bin* [43]	5
DE/*c2b*/1/*bin* [43]	4.17
DE/*rand*/1/*bin* [43]	2.17
CSO-C [1]	3.33
CSO-M (proposed)	1.83
CSO-M/τ/λ (proposed)	**1.5**

Fig. 2 reveals the convergence of the obtained fitness values vs FEVs. Different methods may consume a different number of FEVs in each generation. Nevertheless, the search budget is controlled by the given FEVs but not the generation, and thus each method is evaluated under the same criterion. It can be seen from Fig. 2 that the proposed modified CSO shows significant improvements in terms of the obtained fitness value in Griewank, Rosenbrock, and Weierstrass functions. The proposed strategies add on the searching accuracy but sometimes delay the convergence to larger FEVs. CSO-M and CSO-M/τ/λ outperforms CSO-C in terms of the accuracy in 80% of the test functions. Moreover, it is observable in Rastrigin, Griewank, Sphere, and Rosenbrock functions that CSO-M/τ/λ has a slower convergence speed comparing to CSO-M and CSO-C. We expect that the higher accuracy is contributed by the modified tracing mode because the swarm has higher degree of freedom to move and is not limited by the momentum. Moreover, the delay of convergence is caused by the variable *mode ratio* and the focus boost strategy. The variable *mode ratio* gradually puts less contribution to the global search and thus the whole search characteristic moves from high global exploration to high local exploitation.

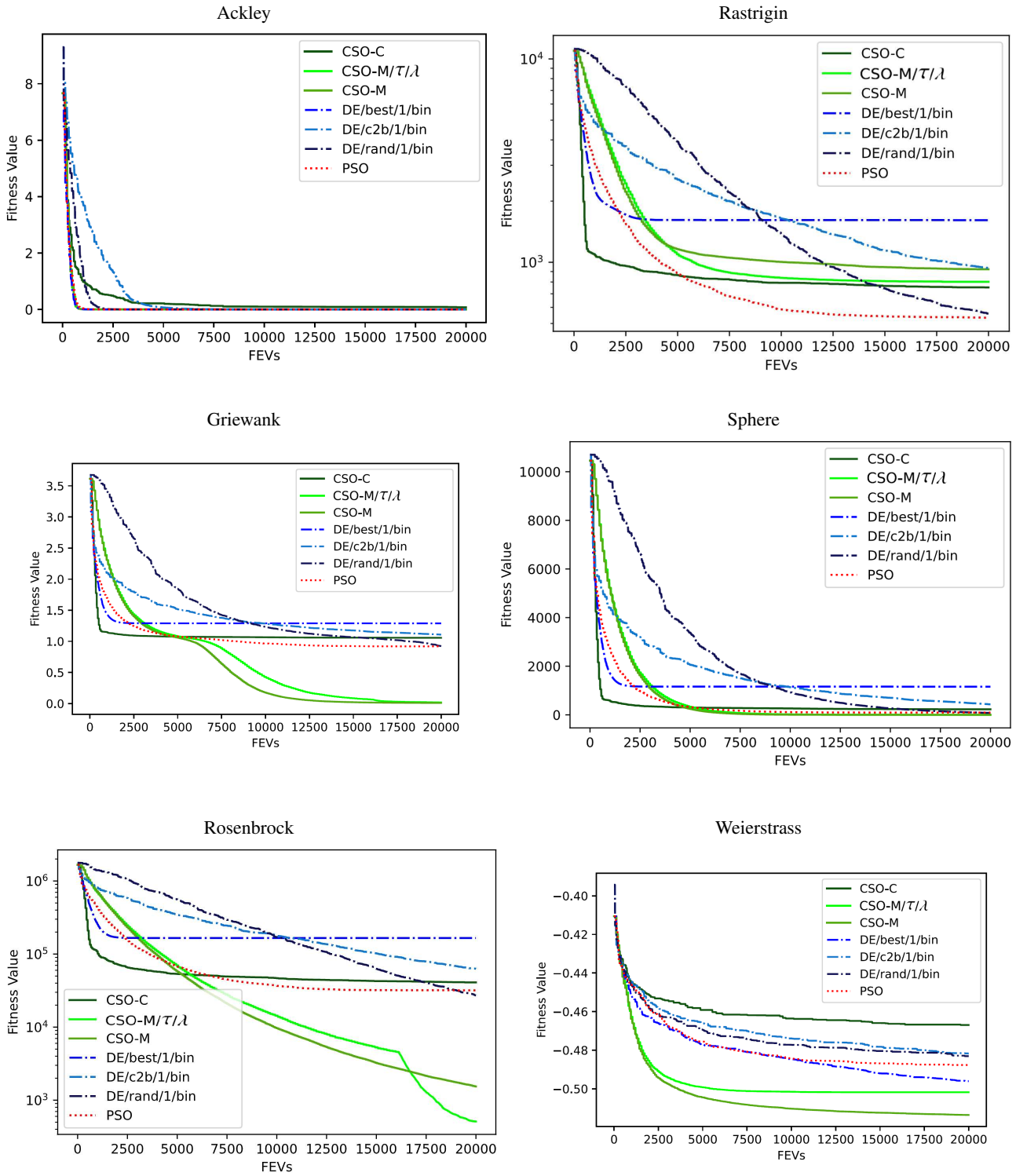

Figure 2. Fitness value convergence vs function evaluations.

5. Conclusions and future work

In this work, we propose a modification on the tracing mode process, the variable *mode ratio* with a control strategy, and a focus boost search strategy for CSO. These three proposed elements make contributions to improve the original CSO algorithm. Experimental results on six test functions from the CEC competition are used in the experiments to evaluate the search accuracy of algorithms. The representative fitness values are calculated by the average of test results in thirty independent runs. The average rank from all experiments is used to evaluate all the methods tested. The experimental results indicate that the proposed modification on the tracing mode shows significant improvement in most test functions compared to the original CSO.

By observing the convergence curves, we conclude that the modified tracing mode offers higher degree of freedom for the swarm to move in the solution space and thus the global exploration ability is stronger than the original CSO. Based on the convergence curves, we find that the original CSO converges much faster than other methods, but the methods have slower convergence speed may gradually discover better solutions than the fast convergence one. The variable *mode ratio* strategy allows the algorithm to balance the global exploration and local exploitation ability by changing the proportion of swarm to be updated in different modes. The focus boost search strategy forces the algorithm to put the remaining resource (in this case, the remaining FEVs) on the most promising individual and refine the solution with only local exploitation when approaching to the end of the search. With the enhancement of the global exploration capacity, the local exploitation power becomes weaker. Thus, CSO with the modified tracing mode converge slower than the original CSO. Nevertheless, the methods having the slower convergence speed may gradually discover better solutions than the fast convergence one. In a sense, this is in analogy to the simulated annealing which reduces the temperature to low near the end of the search in converging to a near-optimal solution. When our proposed method is used in practice, we suggest to tune the variable *mode ratio* (τ) from a larger value toward a smaller value so that the method would have strong global search capacity in the beginning and gradually focus on the local search when it approaches the end of the search. Using γ and δ as the initial and the terminal value of τ and make it linearly decrease along the generation is capable of producing a satisfactory result. Moreover, setting λ to the same value as δ is also recommended to simplify the user-defined process.

The current work focuses on single task optimization. In the future, we plan to investigate and expand CSO's functionality to solve the constrained optimization, multitasking optimization problems and its behavior when encountering the knowledge reuse from other tasks in the asynchronous environment.

Acknowledgments

We would like to thank the constructive feedback provided by the reviewers. This work was performed on the OzSTAR national facility at Swinburne University of Technology. The OzSTAR program receives funding in part from the Astronomy National Collaborative Research Infrastructure Strategy (NCRIS) allocation provided by the Australian Government. This work is supported by the National Natural Science Foundation of China (No. 62172095), the Natural Science Foundation of Fujian Province (No. 2020J01875).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. S. C. Chu, P. W. Tsai, J. S. Pan, Cat Swarm Optimization, PRICAI 2006: Trends in Artificial Intelligence, Pacific Rim International conference on Artificial Intelligence, *Lect. Notes Comput. Sc.*, **4099** (2006), 854–858.
2. D. Debnath, R. Das, P. Pakray, Extractive Single Document Summarization Using Multi-objective Modified Cat Swarm Optimization Approach: ESDS-MCSO, *Neural Computing and Applications*, 2021. online, doi:10.1007/s00521-021-06337-4
3. J. Huang, P. G. Asteris, S. M. K. Pasha, A. S. Mohammed, M. Hasanipanah, A New Auto-tuning Model for Predicting the Rock Fragmentation: A Cat Swarm Optimization Algorithm, *Engineering with Computers*, 2020. online, doi:10.1007/s00366-020-01207-4
4. A. M. Ahmed, T. A. Rashid, S. A. M. Saeed, Dynamic Cat Swarm Optimization algorithm for backboard wiring problem, *Neural Comput. Appl.*, **33** (2021), 13981–13997. doi:10.1007/s00521-021-06041-3
5. P. W. Tsai, J. S. Pan, S. M. Chen, B. Y. Liao, Enhanced parallel Cat Swarm Optimization based on the Taguchi Method, *Expert Syst. Appl.*, **39** (2012), 6309–6319. doi:10.1016/j.eswa.2011.11.117
6. V. I. Skoullis, I. X. Tassopoulos, G. N. Beligiannis, Solving the high school Timetabling Problem using a hybrid Cat Swarm Optimization based algorithm, *Appl. Soft Comput.*, **52** (2017), 277–289. doi:10.1016/j.asoc.2016.10.038
7. K. Balaji, P. S. Kiran, M. S. Kumar, An energy efficient load balancing on cloud computing using adaptive Cat Swarm Optimization, *Materialstoday: proceedings*, In Press, 2020. doi:10.1016/j.matpr.2020.11.106
8. A. Sarswat, V. Jami, R. M. R. Guddeti, A novel two-step approach for overlapping community detection in social networks, *Soc. Netw. Anal. Min.*, **7** (2017), 1–11. doi:10.1007/s13278-017-0469-7
9. N. Kanwar, N. Gupta, K. R. Niazi, A. Swarnkar, Improved Cat Swarm Optimization for simultaneous allocation of DSTATCOM and DGs in distribution systems, *J. Renew. Ener.*, **2015** (2015), 1–10. doi:10.1155/2015/189080
10. J. Li, M. Gao, J. S. Pan, S. C. Chu, A parallel compact Cat Swarm Optimization and its application in DV-Hop node localization for wireless sensor network, *Wirel. Netw.*, **27** (2021), 2081–2101. doi:10.1007/s11276-021-02563-9
11. X. Nie, W. Wang, H. Nie, Chaos Quantum-Behaved Cat Swarm Optimization algorithm and its application in the PV MPPT, *Comput. Intel. Neurosc.*, **2017** (2017), 1–11. doi:10.1155/2017/1583847
12. P. Mohapatra, S. Chakravarty, P. K. Dash, Microarray medical data classification using kernel ridge regression and modified Cat Swarm Optimization based gene selection system, *Swarm Evol. Comput.*, **28** (2016), 144–160. doi:10.1016/j.swevo.2016.02.002

13. H. Siqueira, C. Santana, M. Macedo, E. Figueiredo, A. Gokhale, C. Bastos-Filho, Simplified Binary Cat Swarm Optimization, *Integr. Comput-Aid. E.*, **28** (2021), 35–50. doi:10.3233/ICA-200618

14. M. Gomathy, Optimal feature selection for speech emotion recognition using enhanced Cat Swarm Optimization algorithm, *Int. J. Speech Technol.*, **24** (2021), 155–163. doi:10.1007/s10772-020-09776-x

15. H. Singh, Y. Kumar, A neighborhood search based Cat Swarm Optimization algorithm for clustering problems, *Evol. Intell.*, **13** (2020), 593–609. doi:10.1007/s12065-020-00373-0

16. D. Yan, H. Cao, Y. Yu, Y. Wang, X. Yu, Single-Objective/Multiobjective Cat Swarm Optimization clustering analysis for data partition, *IEEE T. Autom. Sci. Eng.*, **17** (2020), 1633–1646. doi:10.1109/TASE.2020.2969485

17. M. Rao, N. K. Kamila, Cat Swarm Optimization based autonomous recovery from network partitioning in heterogeneous underwater wireless sensor network, *Int. J. Syst. Assur. Eng.*, **12** (2021), 480–494. doi:10.1007/s13198-021-01095-x

18. H. Sikkandar, R. Thiyagarajan, Deep learning based facial expression recognition using improved Cat Swarm Optimization, *J. Amb. Intel. Hum. Comp.*, **12** (2021), 3037–3053. doi:10.1007/s12652-020-02463-4

19. T. V. Vivek, R. R. Guddeti, A hybrid bioinspired algorithm for facial emotion recognition using CSO-GA-PSO-SVM, *The 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)*, 472–477, 2015.

20. M. Kumar, S. K. Mishra, S. K. Choubey, S. S. Tripathy, D. K. Choubey, D. Das, Cat Swarm Optimization based functional link multilayer preceptron for suppression of Gaussian and impulse noise from computed tomography images, *Curr. Med. Imaging*, **16** (2020), 329–339. doi:10.2174/1573405614666180903115336

21. H. Israa, M. Sabah, Improvement Cat Swarm Optimization for efficient motion estimation, *Int. J. Hybrid Inf. Technol.*, **8** (2015), 279–294. doi:10.14257/ijhit.2015.8.1.25

22. M. Suresh, I. S. Sam, Exponential fractional Cat Swarm Optimization for video steganography, *Multimed. Tools Appl.*, **80** (2021), 13253–13270. doi:10.1007/s11042-020-10395-6

23. X. F. Ji, J. S. Pan, S. C. Chu, P. Hu, Q. W. Chai, P. Zhang, Adaptive Cat Swarm Optimization algorithm and its applications in vehicle routing problems, *Math. Probl. Eng.*, **2020** (2020), 1–14. doi:10.1155/2020/1291526

24. M. F. Sohail, C. Y. Leow, S. H. Won, A Cat Swarm Optimization based transmission power minimization for an aerial NOMA communication system, *Veh. Commun.*, 2021. In Press, doi:10.1016/j.vehcom.2021.100426

25. A. M. Ahmed, T. A. Rashid, S. A. M. Saeed, Cat Swarm Optimization algorithm: A survey and performance evaluation, *Comput. Intel. Neurosc.*, **2020** (2020), 1–20. doi:10.1155/2020/4854895

26. R. R. Ihsan, S. M. Almufti, B. M. S. Ormani, R. R. Asaad, R. B. Marqas, A survey on Cat Swarm Optimization algorithm, *Asian J. Res. Comput. Sci.*, **10** (2021), 22–32. doi:10.9734/AJRCOS/2021/v10i230237

27. D. Li, W. Guo, A. Lerch, Y. Li, L. Wang, Q. Wu, An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization, *Swarm Evol. Comput.*, **60** (2021). doi:10.1016/j.swevo.2020.100789

28. baeldung, *P, NP, NP-Complete and NP-Hard Problems in Computer Science,* Available from: https://www.baeldung.com/cs/p-np-np-complete-np-hard

29. J. S. Pan, L. Kong, T. W. Sung, P. W. Tsai, S. Vaclav, A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set, *J. Internet Technol.*, **19** (2018), 1111–1118.

30. X. Xue, C. Jiang, H. Wang, P. W. Tsai, G. Mao, H. Zhu, An improved multi-objective evolutionary optimization algorithm with inverse model for matching sensor ontologies, *Soft Comput.*, **25** (2021), 12227–12240.

31. P. W. Tsai, J. S. Pan, S. M. Chen, B. Y. Liao, S. P. Hao, Parallel Cat Swarm Optimization, *2008 International Conference on Machine Learning and Cybernetics (ICMLC)*, **6** (2008), 3328–3333. doi:10.1109/ICMLC.2008.4620980

32. B. Santosa, M. K. Ningrum, Cat Swarm Optimization for clustering *2009 International Conference of Soft Computing and Pattern Recognition*, 2009, 54–59. doi:10.1109/SoCPaR.2009.23

33. Y. Sharafi, M. A. Khanesar, M. Teshnehlab, Discrete binary Cat Swarm Optimization algorithm *2013 3rd International Conference on Computer, Control & Communication (IC4)*, 2013, 1–6. doi:10.1109/IC4.2013.6653754

34. M. Orouskhani, Y. Orouskhani, M. Mansouri, M. Teshnehlab, A novel Cat Swarm Optimization algorithm for unconstrained optimization problems, *Int. J. Inf. Technol. Comput. Sci.*, **5** (2013), 32–41.

35. P. M. Pradhan, G. Panda, Solving multiobjective problems using cat swarm optimization *Expert Syst. Appl.*, **39** (2012), 2956–2964. doi:10.1016/j.eswa.2011.08.157

36. M. Orouskhani, M. Mansouri, M. Teshnehlab, Average-inertia weighted Cat Swarm Optimization *International Conference in Swarm Intelligence*, 2011, 321–328.

37. D. Bingham, S. Surjanovic, Virtual Library of Simulation Experiments: Test Functions and Datasets, 2013. Available from: https://www.sfu.ca/~ssurjano/ackley.html

38. X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, Benchmark Function for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization, 2013. Available from: https://titan.csit.rmit.edu.au/~e46507/cec13-lsgo/competition/cec2013-lsgo-benchmark-tech-report.pdf

39. R. M. Storn, K. V. Price, Differential evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997), 314–359. doi:10.1023/A:1008202821328

40. S. C. Chu, P. W. Tsai, Computational intelligence based on the behavior of Cats, *Int. J. Innov. Comput. I.*, **3** (2007), 163–173.

41. C. Jin, P. W. Tsai, A. K. Qin, A study on knowledge reuse strategies in multitasking differential evolution, *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, doi:10.1109/CEC.2019.8790102.

42. J. Kennedy, R. Eberhart, Particle swarm optimization, *ICNN'95-International Conference on Neural Networks*, 1995, doi:10.1109/ICNN.1995.488968

43. A. K. Qin, X. Li, Differential Evolution on the CEC-2013 Single-Objective Continuous Optimization Testbed, *2013 IEEE Congress on Evolutionary Computation (CEC'13)*, 2019, 1099–1106. doi:10.1109/CEC.2013.6557689.

AIMS Press