Received: May 27, 2023. Revised: July 18, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Two-Stage Crow Search Algorithm for Solving Optimization Problems

Abdalla El-Dhshan[1]* **Hegazy Zaher[1]** **Naglaa Ragaa[1]**

[1]*Faculty of Graduate Studies for Statistical Research, Cairo University, Egypt*
* Corresponding author's Email: abdalla.mohamed.master@gmail.com

Abstract: This paper proposes a novel metaheuristic technique called two-stage crow search algorithm (TS-CSA). The TS-CSA extends the original crow search algorithm (CSA), that mimics the intelligent foraging behavior of crows. TS-CSA integrates a two-stage search process to improve the CSA in three aspects. First, the two search stages are designed to balance the algorithm's global search and local search capabilities. Second, TS-CSA incorporates a "leaders group" to enable high-quality individuals to guide other individuals and perform global searches by expanding the search space using the best individuals. Third, TS-CSA proposes a new local search strategy for fine-tuning the individuals of the first stage. The proposed TS-CSA and the original CSA are evaluated alongside other metaheuristic algorithms: Grey wolf optimizer (GWO), marine predators algorithm (MPA), and pelican optimization algorithm (POA) using a well-known benchmark of thirteen optimization functions. These functions are commonly employed to assess optimization algorithms due to their complex landscapes and multiple local optima. The results show that the TS-CSA outperforms all competing algorithms, including the original CSA, in terms of convergence and solution quality across all two categories of optimization functions. Furthermore, the proposed algorithm shows potential for solving complex optimization problems across various domains, making it a valuable tool for practitioners and researchers in related optimization fields.

Keywords: Metaheuristic algorithm, Crow search algorithm, Two-stage search strategy, Optimization problems.

1. Introduction

Optimization is an essential task in many areas of science, engineering, and technology, ranging from machine learning to finance and engineering design [1]. In recent years, metaheuristic algorithms have become increasingly popular for solving complex optimization problems due to their ability to handle high-dimensional, non-linear, and non-convex search spaces [2, 3]. In recent times, a multitude of metaheuristic algorithms have been proposed, with many of them being metaphor-based. It is common for the behaviors of birds and animals to serve as inspiration when designing such metaheuristic algorithms. Examples of metaheuristic algorithms that mimic birds and animal behaviors are firefly algorithm (FA) which emulates the bioluminescent communication and attraction behavior of fireflies [4], artificial bee colony (ABC) that mimics the foraging behavior of honey bees, utilizing their collective intelligence [5], grey wolf optimizer (GWO) which models the social hierarchy and hunting behavior of grey wolves [6], cuckoo search (CS) a technique of leverages the unique egg-laying strategy of some cuckoo species which known as brood parasitism [7], pelican optimization algorithm (POA) simulate the natural hunting behavior of pelicans [8]. In this algorithm, search agents represent pelicans that seek food sources, marine predators algorithm (MPA) mimics the hunting strategies of various marine predators and prey [9], and the gorilla troops optimizer (GTO)[10] simulates the organized social dynamics and collaborative interactions of gorilla groups. Metaheuristic algorithms have demonstrated their effectiveness in solving problems in both theoretical test functions and practical applications [11]. Although, there is no single algorithm that can serve as a general solution to all optimization problems. As the performance of optimization algorithms depends on the problem structure, some optimizers may have limitations such

as slow convergence speed and weak ability to avoid local extreme points. Consequently, the no free lunch (NFL) theorem motivates researchers to develop optimization algorithms that are more efficient and effective in solving both present and future optimization problems [12].

The crow search algorithm (CSA) is a relatively new population-based metaheuristic algorithm inspired by the foraging behavior of crows [13]. During its optimization process, each crow in the population follows another randomly selected crow to locate its hidden food source. The awareness probability value dictates if the target crow becomes aware of being followed. In case of awareness, the target crow moves to a random location to safeguard its food source. Simultaneously, the following crow adapts its position based on the target crow's location and flight length, which influences the search distance. This continuous process establishes a balance between exploration and exploitation in the search space, ultimately leading to the discovery of improved solutions [14].

Due to the simplicity and flexibility of the algorithm, several variations have emerged, such as binary CSA, improved CSA, enhanced CSA, multi-strategies CSA, and hybrid CSA. The Binary CSA adapted to handle binary decision variables, by making it suitable for combinatorial optimization problems [15, 16]. In improved CSA, the algorithm parameters are used to improve the CSA. In [17] incorporated dynamic awareness probability and modified the flight length parameter to improve the algorithm's performance. In a separate study, [14] proposed another improved version of CSA, which integrates a weight coefficient, a global best position, and a spiral search mechanism. Additionally, this version employs Gaussian variation to decrease the likelihood of the algorithm getting stuck in local optima, further enhancing its optimization capabilities.

In the modified CSA version, [18] proposed a modification by relocating the new positions of the population based on the position of the global best and the mean of the population. In [19] proposed a modification to improve its exploitation capability. They altered the reproduction process of generating new solutions by incorporating the best global solution from the corresponding iteration. In [20] modified the *AP* fixed value to be linearly decreased between minimum and maximum values during the algorithm iterations to enhance global exploration. Additionally, the generalized Pareto probability density function is used to adjust the flight length value, increasing the search process's diversity. The Hybrid CSA version combined the advantages of two or more methods. Such as CSA hybrid with gray wolf optimizer [21] and hybrid particle swarm optimization [15].

Based on the aforementioned review, it is evident that many existing studies have focused on improving the crow search algorithm (CSA), but certain limitations remain. These drawbacks include the reliance on a single type of search guidance for individuals and search stages, limited adaptability to complex and high-dimensional problems, and an imbalance between global exploration and local exploitation. Addressing these challenges could further enhance the effectiveness of CSA in solving diverse optimization problems.

The goal of this work is to propose a novel metaheuristic called the two-stage crow search algorithm (TS-CSA). This algorithm is designed as an enhanced version of the original CSA. The main contributions of this work are outlined below:

1) The optimization process is divided into two stages: an initial exploitation and exploration stage and posterior exploitation stage. This division helps improve the algorithm's robustness and enhances the quality of the final solution.
2) A "leaders group" is introduced to guide the search process toward high-quality individuals. The positions of the crows are updated based on the fittest solutions (leaders group) in the population base, which ensures that the algorithm converges rapidly.
3) The flight length value is dynamically generated using a random value, rather than being fixed. This flexibility improves the exploration capability of the algorithm.
4) A new local search strategy is added to fine-tune the solution quality, further enhancing the performance of the algorithm.

The subsequent sections of this paper are arranged as follows. In section 2, the overview of the CSA is introduced. In section 3, the proposed TS-CSA algorithm is presented in detail. In section, 4, the experimental implications and results have discussed. Finally, in Section 5, The conclusions and future work are provided.

2. Crow search algorithm (CSA)

The crow is an extremely intelligent bird. Scientists have indicated that the crow's intelligence is due to brain size significantly larger than expected for their body size. Crows possess remarkable cognitive abilities such as long-term memory for

N	Population size
$iter\_Max$	Maximum number of iterations
$iter$	Iteration number *iter*
i, j	Crows in the population
d	Dimension number of search space
x_i^{iter}	Solution of $crow_i$ at iteration *iter*
$x^{L,iter}$	The selected leader crow
$Global\_Best$	Best solution so far
r_i, r_j, r_L	Random numbers
P_i	Position of $Crow_i$
fl	flight length
fl_{s1} *and* fl_{s2}	flight length of stage 1 and stage 2
AP	awareness probability
$Leaders\_group$	the top 50% of the crows

facial recognition, advanced problem-solving skills, sophisticated communication abilities, self-awareness, and even tool-making abilities, which enable them to thrive in their environment [22]. Based on this premise, crow search algorithm (CSA) is presented as a new population-based algorithm [13].

The algorithm's inspiration came from the intelligent behavior of crows, which live in the form of a flock and memorize the position of their hiding places. They also follow each other to carry out theft and protect their caches from being pilfered. Below are the annotations used in this work.

In the CSA algorithm, the optimization process of the algorithm involves a population size and a maximum number of iterations. In each iteration, the position of $crow_i$ in the dimensional search space is represented by x_i^{iter}, where *i*=1,2,..., *N*, and $iter$ =1,2,..., g_{max}. The best-hidden food position obtained by $crow_i$ up to that iteration is represented by $p^{i,iter}$. To find the hidden food position of $crow_j$, $crow_i$ follows $crow_j$ in each iteration. The best solution found until the current iteration is determined by $Global\_Best^{iter}$. Finally, the search space vector can be represented by $x_i^{iter} = [x_1^{iter}. x_2^{iter}. \ldots . x_d^{iter}]$.

Updating crows' position in CSA has two states. In State 1, $crow_j$ is unaware that $crow_i$ is following it. Consequently, $crow_i$ can steal the hiding place of $crow_j$. In State 2, $crow_j$ is aware that $crow_i$ is following it. To protect its cache from being pilfered, $crow_j$ will deceive $crow_i$ by moving to another position in the search space. Eq. (1) summarizes the two states for updating the position of crows as follows:

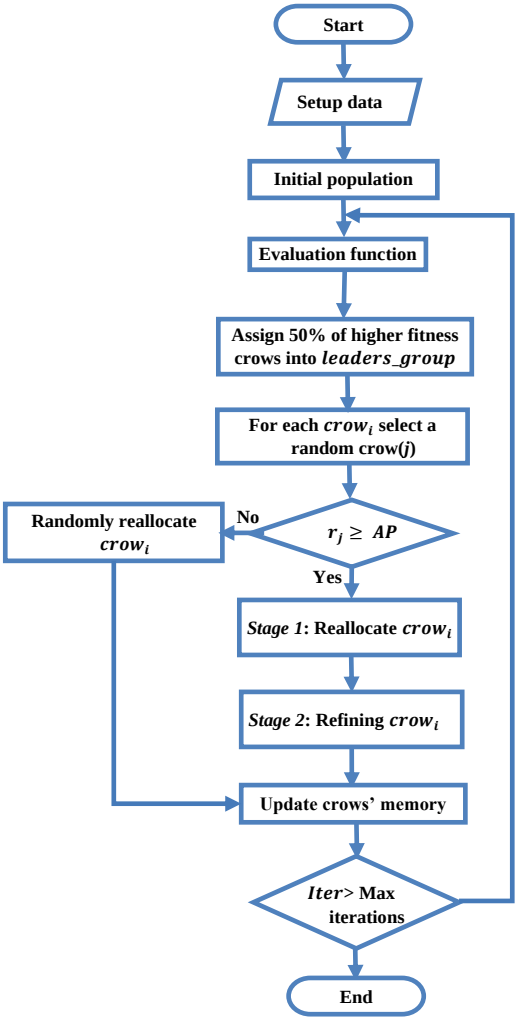

Figure. 1 Flowchart of a two stages crow search algorithm

$$x_i^{iter+1} = \begin{cases} x_i^{iter} + r_i \times fl_i^{iter} \times \left(p_j^{iter} - x_i^{iter}\right) & r_j \geq AP_j^{iter} \\ a\ random\ possion & otherwise \end{cases} \quad (1)$$

Where x_i^{iter+1} is the next position for $crow_i$ at iteration $iter$. r_i and r_j values are generated with a uniform distribution between 0 and 1. p_j^{iter} is the current position of $crow_j$. fl_i^{iter} is the flight length of $crow_i$ at iteration $iter$, while AP_j^{iter} is the awareness probability of $crow_j$ at iteration $iter$. Simultaneously, the position memory of each crow was updated using Eq. (2).

$$p_i^{iter+1} = \begin{cases} x_i^{iter+1} & if\ f\left(x_i^{iter+1}\right) > f\left(x_i^{iter}\right) \\ p_i^{iter+1} & otherwise \end{cases} \quad (2)$$

$$x^{i,iter+1} = \begin{cases} x^{i,iter} + r_i \times fl_{s1}^{i,iter} \times \left(x^{L,iter} - x^{i,iter}\right) & if\ r_L \geq AP^{L,iter} \\ a\ random\ possion & otherwise \end{cases} \quad (5)$$

Algorithm 1: pseudo code of TS-CSA

Input number of problem dimension (*d*), Maximum of iteration ($iter\_Max$), population size (N), and awareness probability (*AP*).
Initialize the memory for each crow in the population matrix with the size of Pop_Size and evaluate the objective function.
while $iter < iter\_Max$
Sort crows in descending orders based on the evaluation values.
//Stage 1
Set $Leaders\_Group$ with the top 50% of the sorted crows.
for $i < N$
Choose $crow\ j$ randomly from the list of crows in $Leaders\_group$.
If $rand_j \geq AP$
Assign the fl_{s1} with a uniform random number from interval [-1,1].
Update the position of $crow_i$ by Eq. (3).
Else
Update a random position for $crow_i$.
end if
Maintain the feasibility of the new position $crow_i$.
Evaluate the objective function of $crow_i$.
//Stage 2
Assign the fl_{s2} with a uniform random number from interval]1,2].
Update New_crow_i by Eq. (4).
Maintain the feasibility of the new position $crow\ i$.
Evaluate the objective function of New_crow_i.
If fitness(New_crow_i) > fitness($crow_i$)
allocate $New\_crow_i\ to\ New\_crow_i$.
end if
Update memory of $crow_i$.
Update the best crow.
end for
end while
Return the global best crow.

3. Two stages crow search algorithm (TS - CSA)

This section introduces in detail the model of TS-CSA. As previously mentioned, CSA has several drawbacks, such as premature convergence, being trapped into local optimum, and a low convergence rate. The primary drawback of CSA is that its control parameters, *AP* and fl, are fixed, which hinders the algorithm's ability to explore and exploit the search space optimally [23]. Using fixed values for these parameters may be effective in one stage of the optimization process but not in another throughout the optimization process [18].

The proposed algorithm, TS-CSA, aims to overcome the drawbacks and enhance the performance of the CSA. Thus, two stages of the optimization process are proposed. During the first stage, the flight length (fl) plays an essential role to allow the algorithm to perform a local search where the fl_{s1} will be a random value between 0 and 1.

Generally, adding guiding movement based on the individual whose has high fitness value is improving the performance of the search process [24]. From this aspect, fifty percent of the population size is grouped as leaders. Crows in the leaders group are assigned based on their fitness values. Then, each crow in the overall population updates its position based on choosing a random crow of leaders group using Eq. (3).

$$x^{i.iter+1} = x^{i,iter} + r_i \times fl_{s1}^{i,iter} \times \left(x^{leader,iter} - x^{i,iter}\right) \quad (3)$$

This contribution significantly improves the exploitation capability of the algorithm by guided toward the best solutions. On the other hand, another contribution represents in the second stage: the flight length value (fl_{s2}) will be greater than 1, based on a random number (fl_{s2}:]1,2]). As fl with large value guides to global search which insure the diversity of the search space. Moreover, the second stage implements a local search by fine-tuning the position of the generated solution at the first stage based on Eq. (4). The coordination of the two stages emphasizes the balance between the exploration and the exploitation capabilities of the proposed TS-CSA.

$$New\_crow_i = crow_i + fl_{s2} * crow_i \quad (4)$$

Fig. 1 and Algorithm 1 provide an overview of the TS-CSA algorithm. The proposed algorithm can be described in more detail as follows:

Step 1: Parameters declaration, TS-CSA algorithm's parameters include problem dimension (*d*), Max number of iterations ($iter\_Max$), the population size of crows (*N*), and awareness probability (*AP*).
Step 2: Initialize random position and memory.
Step 3: Calculate the fitness value for each crow based on the objective function.
Step 4: Sort the population according to fitness values,

Table 1. Description of unimodal benchmark functions

Function	Range	Dimension	Best known solution
$\mathbf{F1(x)} = \sum_{i=1}^{n} x_i^2$	[-100, 100]	30	0
$\mathbf{F2(x)} = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]	30	0
$\mathbf{F3(x)} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	[-100, 100]	30	0
$\mathbf{F4(x)} = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	[-100, 100]	30	0
$\mathbf{F5(x)} = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	[-30, 30]	30	0
$\mathbf{F6(x)} = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	[-100, 100]	30	0
$\mathbf{F7(x)} = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	[-1.28, 1.28]	30	0

Table 2. Description of multimodal benchmark functions

Function	Range	Dimension	Best known solution
$\mathbf{F8(x)} = \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]	30	-12569.5
$\mathbf{F9(x)} = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	[-5.12, 5.12]	30	0
$\mathbf{F10(x)} = 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	[-32, 32]	30	0
$\mathbf{F11(x)} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$	[-600, 600]	30	0
$\mathbf{F12(x)} = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_{i-1})^2[1 + 10\,X\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u((x_i, 10, 100.4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i > -a \end{cases}$	[-50, 50]	30	0
$\mathbf{F13(x)} = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + 10\sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u((x_i, 5, 100.4)$	[-50, 50]	30	0

Table 3. The experimental results of unimodal optimization functions

Function	Average Fitness Value				
	TS-CSA	**POA** [8]	**CSA** [13]	**MPA** [9]	**GWO [5]**
F1	$\mathbf{1.623\times10^{-31}}$	8.013×10^{-16}	2.566×10^{1}	1.014×10^{-1}	1.434×10^{-2}
F2	$\mathbf{5.500\times10^{-22}}$	7.605×10^{-9}	2.357	1.023×10^{-1}	2.481×10^{-2}
F3	$\mathbf{2.179\times10^{-36}}$	5.459×10^{-15}	5.522×10^{3}	2.464×10^{2}	2.485×10^{2}
F4	$\mathbf{7.515\times10^{-24}}$	5.415×10^{-9}	4.235×10^{-1}	4.930×10^{-1}	1.292
F5	$\mathbf{1.973\times10^{-4}}$	2.889×10^{1}	4.014×10^{3}	3.029×10^{1}	3.358×10^{1}
F6	$\mathbf{3.201\times10^{-7}}$	4.913	5.251×10^{1}	1.584	2.971
F7	$\mathbf{4.285\times10^{-4}}$	9.457×10^{-4}	4.497×10^{-2}	9.229×10^{-3}	2.125×10^{-2}

Table 4. The experimental results of multimodal optimization functions

Function	Average Fitness Value				
	TS-CSA	**POA** [8]	**CSA** [13]	**MPA** [9]	**GWO [5]**
F8	**-1.257×10^{4}**	-6.385×10^{3}	-1.224×10^{4}	-7.157×10^{3}	-5.408×10^{3}
F9	**0.000**	8.281×10^{-8}	1.740×10^{1}	1.440×10^{1}	3.554×10^{1}
F10	**3.171×10^{-14}**	5.021×10^{-9}	1.865	8.219×10^{-2}	2.982×10^{-2}
F11	**0.000**	4.219×10^{-16}	1.082	2.429×10^{-1}	9.675×10^{-2}
F12	**1.024×10^{-10}**	3.901×10^{-1}	7.994×10^{-1}	7.064×10^{-2}	5.174×10^{-1}
F13	**2.035×10^{-9}**	2.834	1.547	8.686×10^{-1}	2.098

and store the top 50% of the sorted crows inside the $Leaders_group$s et.

Step 5: Regenerate a new position for $crow_i$ by using the proposed two stages as follows:

Stage 1: Randomly choose a $x^{L,iter}$ from the set of $Leader\_group$ at iteration $iter$. In case of the AP is less than the generated random number r_L, then the position will update randomly. The two stages of stage 1 can be summarized by Eq. (5).

Stage 2: Allocate the flight length of the second stage fl_{s2} by a uniform random number between -1 and 1. Then, refine the generated solution produced by stage 1 by using Eq. (4).

Step 6: Evaluate the fitness of the new passions of $crow_i$ *and* New_crow_i.

Step 7: Update the memory of the ith crow by the best greatest *fitness value* of $crow_i$ and New_crow_i.

Step 8: Repeat Steps 4 to 7 until get the maximum number of iterations ($iter\_Max$) and stop.

4. Experiments result

This section presents the performance evaluation of the proposed TS-CSA in solving theoretical optimization problems, a benchmark of thirteen testing functions including unimodal and multimodal types [25] uses to validate the proposed algorithm with other metaheuristics.

The testing optimization functions are divided into two groups unimodal, and multimodal. In Table 1, functions F1 to F7 represent the unimodal group which has only one optimum value. Typically, the unimodal type is used to ensure that the algorithm can explore the search space and has a reasonable convergence rate. Table 2 represents the second set of functions for testing multimodal types F8 to F13. The functions have several local minima which increases exponentially based on the number of the solution dimension. This type of optimization uses to evaluate the algorithm capabilities of exploiting and avoiding local optima and converge to the near-global optimum.

In the first experiment, TS-CSA was compared to four other metaheuristic algorithms: CSA [13], GWO [6], MPA [9], and POA [8]. The reasons for selecting these algorithms are as follows: all algorithms are population-based. They are inspired by animals' or birds' behavior. CSA was included as a baseline for comparison since TS-CSA is an improvement over CSA. GWO, a well-known algorithm that predates CSA, was also included. MPA and POA were selected as more recent metaheuristic algorithms. Furthermore, the source code of all algorithms is publicly available, which facilitated the implementation of the required experiments.

The parameters are configured as follows. The population size is 30. The maximum number of iterations is 100 for all algorithms that were used in the experiment. In CSA, *AP* is 0.1. The *FL is* 1.8. In TS-CSA, *AP* is set 0.1. Every algorithm runs 30 times independently. Octave as an open-source software has been used to design and code all algorithms and execute the experiments. The results are presented in Tables 3 and 4 The best average fitness value is presented in bold text.

Table 3 shows the results of the unimodal optimization type. The results demonstrate the superiority of the proposed TS-CSA across other competing algorithms. The TS-CSA algorithm has the ability to minimize the objective function and achieve the best average fitness value for all functions. Table 4 shows the results of the high-dimensional multimodal optimization type. The results indicate that TS-CSA can obtain the best result in all functions. In addition, TS-CSA can find the global optima in functions F8, F9, and F11.

Figs. 2 and 3 demonstrate the convergence outstanding of the proposed TS-CSA algorithm with other algorithms. TS-CSA significantly outperforms CSA, POA, MPA, and GWO on convergence speed. TS-CSA requires a minimal number of iterations to

Table 5. Relation of the maximum number of iterations and TS-CSA performance

Function	Average Fitness Value		
	Iter_Max (50)	Iter_Max (100)	Iter_Max (150)
F1	3.447×10^{-19}	9.063×10^{-37}	$\mathbf{1.481\times10^{-52}}$
F2	1.329×10^{-10}	1.680×10^{-24}	$\mathbf{1.256\times10^{-33}}$
F3	4.250×10^{-19}	9.196×10^{-47}	$\mathbf{4.252\times10^{-59}}$
F4	6.136×10^{-10}	5.854×10^{-22}	$\mathbf{3.704\times10^{-27}}$
F5	7.793×10^{-2}	$\mathbf{7.344\times10^{-5}}$	1.833×10-4
F6	2.782×10^{-6}	2.639×10^{-8}	$\mathbf{9.172\times10^{-10}}$
F7	6.405×10^{-4}	3.065×10^{-4}	$\mathbf{2.939\times10^{-4}}$
F8	-1.252×10^{4}	$\mathbf{-1.257\times10^{4}}$	$\mathbf{-1.257\times10^{4}}$
F9	**0.000**	**0.000**	**0.000**
F10	4.127×10^{-9}	1.036×10^{-15}	$\mathbf{4.441\times10^{-16}}$
F11	**0.000**	**0.000**	**0.000**
F12	5.748×10^{-8}	2.191×10^{-10}	$\mathbf{7.765\times10^{-11}}$
F13	2.223×10^{-6}	2.820×10^{-9}	$\mathbf{7.084\times10^{-13}}$

Table 6. Relation of population size and TS-CSA performance

Function	Average Fitness Score		
	Population Size=10	Population Size=20	Population Size=30
F1	9.697×10^{-29}	2.860×10^{-31}	$\mathbf{1.665\times10^{-36}}$
F2	1.143×10^{-17}	1.302×10^{-20}	$\mathbf{4.644\times10^{-23}}$
F3	1.317×10^{-27}	3.474×10^{-36}	$\mathbf{1.231\times10^{-40}}$
F4	4.236×10^{-18}	$\mathbf{1.128\times10^{-18}}$	2.679×10^{-16}
F5	1.929	5.779×10^{-3}	$\mathbf{6.629\times10^{-4}}$
F6	1.779×10^{-6}	1.699×10^{-7}	$\mathbf{1.755\times10^{-8}}$
F7	1.063×10^{-3}	4.891×10^{-4}	$\mathbf{4.535\times10^{-4}}$
F8	-1.220×10^{4}	-1.245×10^{4}	$\mathbf{-1.249\times10^{4}}$
F9	**0.000**	**0.000**	**0.000**
F10	1.169×10^{-14}	5.625×10^{-16}	$\mathbf{4.441\times10^{-16}}$
F11	**0.000**	**0.000**	**0.000**
F12	2.782×10^{-8}	7.648×10^{-9}	$\mathbf{6.843\times10^{-10}}$
F13	7.117×10^{-6}	1.342×10^{-8}	$\mathbf{1.111\times10^{-9}}$

converge.

The second experiment assessed to evaluate the relationship between gradually increasing the maximum number of iterations and the performance of TS-CSA. Three values of *Iter_Max* were examined: 50, 100, and 150. The results, displayed in Table 5, reveal a positive correlation between the maximum number of iterations and the algorithm's performance. As the maximum number of iterations increases the fitness value tends to decrease. Moreover, the data exhibits convergence at lower maximum iterations for all functions. The TS-CSA within 50 iterations reaches the global optima in functions F9, and F11 and it obtains near-global optima in other functions.

The third experiment is performed to evaluate the relation between the population size and the algorithm performance. In this simulation, there are three values of population size: 10, 30, and 50.

Table 6 shows that in general, TS-CSA is acceptable to set 30 individuals as population size. TS-CSA finds its best convergence in the highest population size in solving all functions except F4 which reaches the best fitness value for the setup of 20 individuals as population size. Meanwhile, the result obtains in the middle is very near to the high population size for other functions.

The fourth experiment is conducted to evaluate the effectiveness of the awareness probability (*AP*) value on the TS-CSA performance. In this experiment, the *AP* values are set as (0.1, 0.2, 0.3). The output is shown in Table 7.

Table 7. Relation of awareness probability and TS-CSA performance

Function	Average Fitness Score		
	***AP*=0.10**	***AP*=0.20**	***AP*=0.30**
F1	1.851×10^{-40}	3.624×10^{-29}	$\mathbf{1.799\times10^{-42}}$
F2	1.493×10^{-22}	$\mathbf{4.001\times10^{-24}}$	4.349×10^{-18}
F3	3.605×10^{-27}	$\mathbf{1.230\times10^{-39}}$	2.702×10^{-36}
F4	1.017×10^{-21}	$\mathbf{4.733\times10^{-24}}$	1.144×10^{-19}
F5	$\mathbf{6.977\times10^{-4}}$	1.018×10^{-3}	1.216×10^{-3}
F6	$\mathbf{3.137\times10^{-8}}$	3.214×10^{-8}	1.708×10^{-7}
F7	4.120×10^{-4}	$\mathbf{3.765\times10^{-4}}$	6.008×10^{-4}
F8	$\mathbf{-1.257\times10^{4}}$	$\mathbf{-1.257\times10^{4}}$	-1.249×10^{4}
F9	**0.000**	**0.000**	**0.000**
F10	5.625×10^{-16}	$\mathbf{4.441\times10^{-16}}$	$\mathbf{4.441\times10^{-16}}$
F11	**0.000**	**0.000**	**0.000**
F12	1.421×10^{-9}	$\mathbf{1.495\times10^{-10}}$	1.756×10^{-8}
F13	$\mathbf{1.213\times10^{-8}}$	1.367×10^{-7}	2.116×10^{-7}

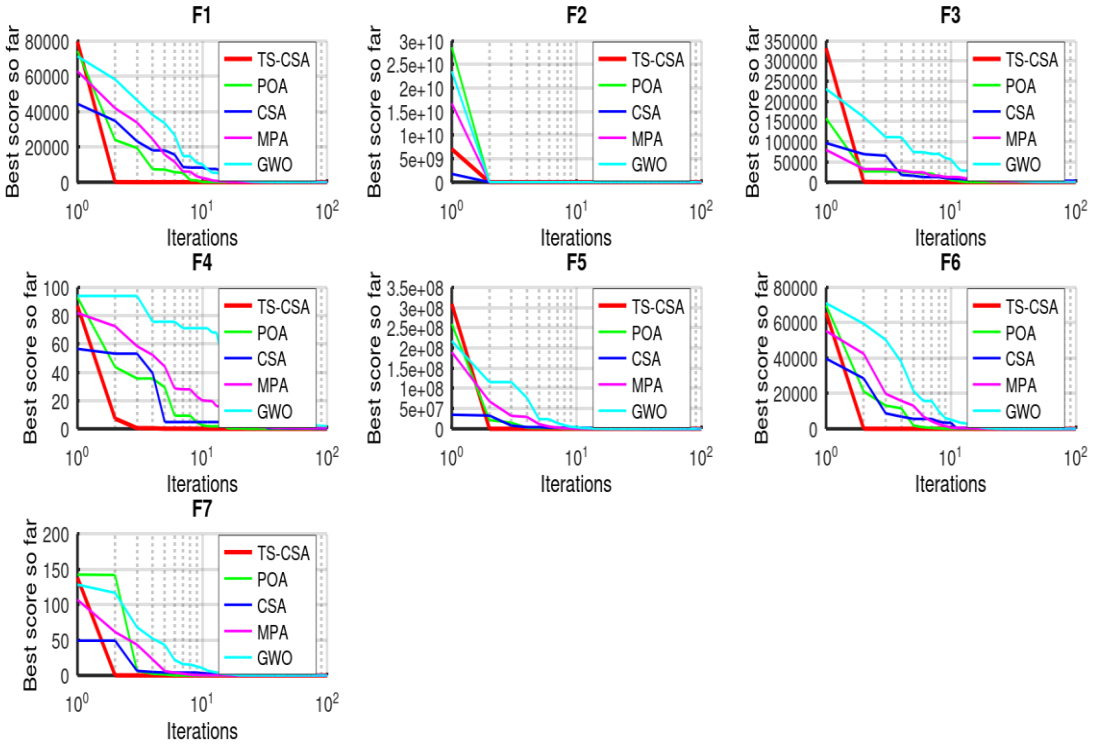

Figure. 2 Solution convergence curve of unimodal functions (F1 to F7)

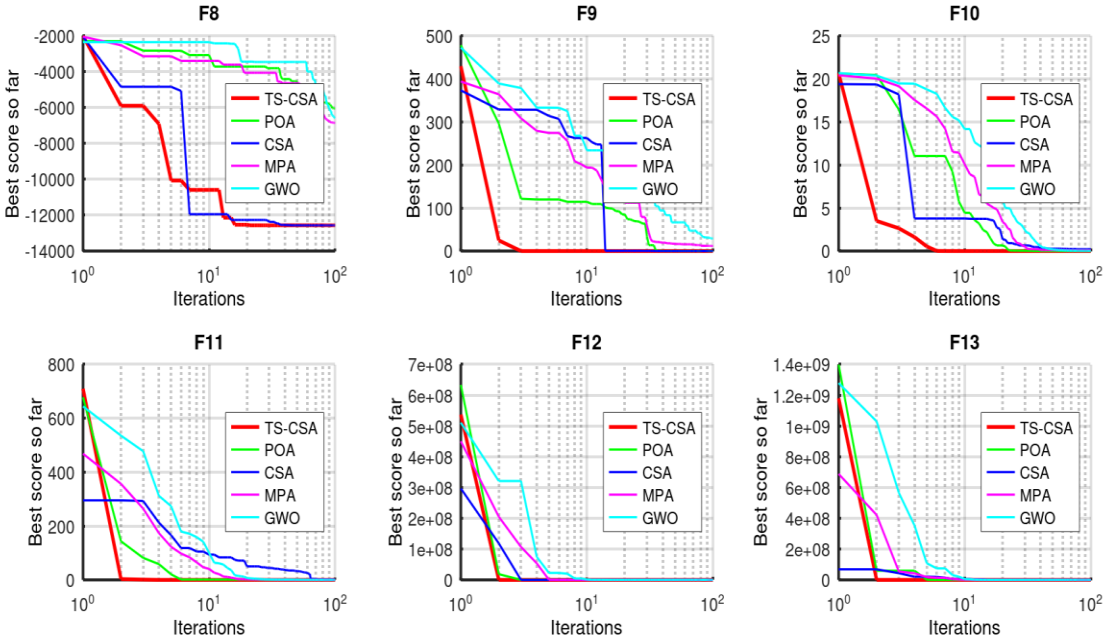

Figure. 3 Solution convergence curve of high-dimensional multimodal functions (F8 to F13)

Table 7 shows that the middle *AP* value is better to select as it improves the performance of TS-CSA in nine functions. While the low *AP* value is getting best in three functions (F5, F6, and F13). In general, the higher AP value does not significantly affect the performance, except in only one function F1.

5. Conclusion

In this paper, we propose a novel variant of the crow search algorithm (CSA) called TS-CSA. TS-CSA model is based on incorporating two stages of the optimization process. The result shows the

superiority of the proposed TS-CSA against other competitor algorithms (CSA, POA, MPA, and GWO) in solving a group of unimodal, and high-dimensional multimodal functions The experiment analysis emphasizes that TS-CSA has a high ability to balance between exploitation, and exploration to perform powerfully in the optimization process. additionally, TS-CSA has the ability of converge to the best solutions faster than other algorithms.

Future research directions can be pursued in several ways. The proposed TS-CSA algorithm can be used to solve a wide range of optimization problems. TS-CSA can be adapted to solve combinatorial optimization problems. Additionally, deep parameters analysis of the proposed TS-CSA may lead to potential improvements and greater efficiency.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The first author was responsible for conducting the background work, conceptualization, methodology development, dataset collection, implementation, result analysis, and comparison, drafting and editing the paper, as well as creating visualizations. The second and third authors provided supervision, reviewed the work, and oversaw the project administration.

References

[1] S. Sharma and V. Kumar, "A Comprehensive Review on Multi-objective Optimization Techniques: Past, Present and Future", *Archives of Computational Methods in Engineering*, Vol. 29, No. 7, pp. 5605–5633, 2022, doi: 10.1007/s11831-022-09778-9.

[2] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey", *Artificial Intelligence Review*, Vol. 52, No. 4, pp. 2191–2233, 2019, doi: 10.1007/s10462-017-9605-z.

[3] L. Abualigah, "Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications", *Neural Computing and Applications*, Vol. 33, No. 7, pp. 2949–2972, 2021, doi: 10.1007/s00521-020-05107-y.

[4] X. S. Yang, "Firefly algorithms for multimodal optimization", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 5792 LNCS, pp. 169–178, 2009, doi: 10.1007/978-3-642-04944-6_14.

[5] D. Karaboga, "An idea based on Honey Bee Swarm for Numerical Optimization", *Technical Report TR06, Erciyes University*, No. TR06, p. 10, 2005, [Online]. Available: http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf

[6] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer", *Advances in Engineering Software*, Vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[7] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems", *Engineering with Computers*, Vol. 29, No. 1, pp. 17–35, 2013, doi: 10.1007/s00366-011-0241-y.

[8] P. Trojovský and M. Dehghani, "Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications", *Sensors*, Vol. 22, No. 3, 2022, doi: 10.3390/s22030855.

[9] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine Predators Algorithm: A nature-inspired metaheuristic", *Expert Systems with Applications*, Vol. 152, No. March, 2020, doi: 10.1016/j.eswa.2020.113377.

[10] B. Abdollahzadeh, F. S. Gharehchopogh, N. Khodadadi, and S. Mirjalili, "Mountain Gazelle Optimizer: A new Nature-inspired Metaheuristic Algorithm for Global Optimization Problems", *Advances in Engineering Software*, Vol. 174, No. 10, pp. 5887–5958, 2022, doi: 10.1016/j.advengsoft.2022.103282.

[11] R. P. Parouha and P. Verma, "State-of-the-Art Reviews of Meta-Heuristic Algorithms with Their Novel Proposal for Unconstrained Optimization and Applications", *Archives of Computational Methods in Engineering*, Vol. 28, No. 5, pp. 4049–4115, 2021, doi: 10.1007/s11831-021-09532-7.

[12] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67–82, 1997, doi: 10.1109/4235.585893.

[13] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm", *Computers and Structures*, Vol. 169, pp. 1–12, 2016, doi: 10.1016/j.compstruc.2016.03.001.

[14] X. Han, Q. Xu, L. Yue, Y. Dong, G. Xie, and X. Xu, "An Improved Crow Search Algorithm Based on Spiral Search Mechanism for Solving Numerical and Engineering Optimization

Problems", *IEEE Access*, Vol. 8, pp. 92363–92382, 2020, doi: 10.1109/ACCESS.2020.2980300.

[15] A. Adamu, M. Abdullahi, S. B. Junaidu, and I. H. Hassan, "An hybrid particle swarm optimization with crow search algorithm for feature selection", *Machine Learning with Applications*, Vol. 6, No. June, p. 100108, 2021, doi: 10.1016/j.mlwa.2021.100108.

[16] S. Laabadi, M. Naimi, H. E. Amri, and B. Achchab, "A binary crow search algorithm for solving two-dimensional bin packing problem with fixed orientation", *Procedia Computer Science*, Vol. 167, pp. 809–818, 2020.

[17] E. Cuevas, E. B. Espejo, and A. C. Enríquez, "A modified crow search algorithm with applications to power system problems", *Studies in Computational Intelligence*, Vol. 822, Springer, 2019, pp. 137–166. doi: 10.1007/978-3-030-11593-7_6.

[18] J. Gholami, F. Mardukhi, and H. M. Zawbaa, "An improved crow search algorithm for solving numerical optimization functions", *Soft Computing*, Vol. 25, No. 14, pp. 9441–9454, 2021.

[19] R. Roy, T. P. Sahu, N. K. Nagwani, and S. Das, "Global best guided crow search algorithm for optimization problems", *Intelligent Algorithms for Analysis and Control of Dynamical Systems*, Springer, 2021, pp. 13–22.

[20] S. Nayeri, R. T. Moghaddam, Z. Sazvar, and J. Heydari, "A heuristic-based simulated annealing algorithm for the scheduling of relief teams in natural disasters", *Soft Computing*, Vol. 26, No. 4, pp. 1825–1843, 2022.

[21] F. Davoodkhani, S. A. Nowdeh, A. Y. Abdelaziz, S. Mansoori, S. Nasri, and M. Alijani, "A new hybrid method based on gray wolf optimizer-crow search algorithm for maximum power point tracking of photovoltaic energy system", *Modern Maximum Power Point Tracking Techniques for Photovoltaic Energy Systems*, pp. 421–438, 2020.

[22] N. J. Emery and N. S. Clayton, "The mentality of crows: Convergent evolution of intelligence in corvids and apes", *Science*, Vol. 306, No. 5703, pp. 1903–1907, 2004, doi: 10.1126/science.1098410.

[23] A. Necira, D. Naimi, A. Salhi, S. Salhi, and S. Menani, "Dynamic crow search algorithm based on adaptive parameters for large-scale global optimization", *Evolutionary Intelligence*, Vol. 15, No. 3, pp. 2153–2169, 2022.

[24] P. D. Kusuma and A. L. Prasasti, "Guided Pelican Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 6, pp. 179–190, 2022, doi: 10.22266/ijies2022.1231.18.

[25] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster", *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp. 82–102, 1999, doi: 10.1109/4235.771163.