Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 21, No. 2, 2026

Adaptive Crow Search Algorithm Based on Population Diversity

Hongran Wang, Nuo Luo

School of Chongqing University of Technology, Chongqing 500113, China

Abstract: As an emerging metaheuristic optimization algorithm, the Crow Search Algorithm (CSA) has received widespread academic attention for its intuitive model and concise parameter settings. However, its inherent limitations, including the tendency to fall into local optima and insufficient convergence accuracy in solving complex optimization problems, have severely restricted its popularization and application in practical engineering scenarios. Aiming at the above shortcomings of the basic CSA, this thesis conducts in-depth and systematic research, proposes two targeted improved algorithms, and systematically verifies their theoretical performance and engineering application value.To address the performance bottleneck of the basic CSA, this thesis proposes an Adaptive Crow Search Algorithm (ACSA) based on population diversity. By constructing an adaptive framework with population diversity as the feedback signal, the algorithm achieves three core improvements. First, a dynamic dual-mode guidance mechanism is designed to intelligently switch between global exploration and local exploitation strategies according to the population diversity index. Second, the Sigmoid function is introduced to realize adaptive adjustment of flight length, enabling the parameter settings to match the real-time search state of the algorithm. Third, the golden sine strategy is adopted to optimize the individual position update rule, which is combined with a reflection boundary handling mechanism to enhance the iterative stability of the algorithm.

Keywords: Crow Search Algorithm; Adaptive Optimization; Population Diversity.

1. Introduction

The core task of optimization theory and algorithms is to find the best solution from among many feasible options under specific constraints. With the continuous improvement in decision-making efficiency and quality requirements across various fields of modern society (such as industrial design, resource allocation, urban planning, etc.), optimization technology has become a key support for promoting technological progress and industrial upgrading. At the same time, the leap in computing power has made it possible to solve complex optimization problems that were previously difficult to handle.

However, with the rapid increase in the complexity of scientific and engineering problems (manifested as high dimensionality, nonlinearity, multimodality, etc.), many practical problems have been proven to belong to NP-hard problems. Traditional exact methods based on gradients or convex optimization often face the dilemma of excessively high computational costs or inapplicability. Against this backdrop, metaheuristic algorithms, as an important component of soft computing, have been widely favored because they do not rely on the strict mathematical properties of problems and are adept at finding satisfactory solutions for complex problems within an acceptable time frame. The appeal of metaheuristic algorithms also comes from their unique advantages: their frameworks are relatively simple and easy to implement, the embedded randomness mechanism helps to escape local optimal traps, and they naturally accommodate whether the objective function is continuous or differentiable.

2. Crow Search Algorithm (CSA)

The Crow Search Algorithm (CSA) is a swarm intelligence optimization algorithm inspired by the intelligent behaviors exhibited by crow populations in nature. Researchers have observed that when crows face larger food resources, they often display highly cooperative and strategic behaviors. Since they cannot carry all the food at once, crows will divide it and transfer it to concealed locations as much as possible to reduce the risk of theft by other crows. In addition, crows possess significant memory and anti-surveillance abilities: they hide their own food while closely observing the behavior of other individuals and tracking their food storage locations; once they detect being followed, they adopt deception strategies, such as flying to random locations, to protect their actual food storage points.

CSA is a metaheuristic optimization algorithm that simulates the intelligent social behaviors of crows storing and stealing food. Its design inspiration comes from the high level of cooperation and strategy displayed by crows during foraging. The algorithm is based on four basic principles:

(1) Crows live in groups and interact socially with one another;

(2) Crows can remember the locations of their own stored food and retrieve it accordingly;

(3) Crows track other individuals to discover their food storage points;

(4) Crows have a certain level of precaution and adopt strategies to protect their food from being stolen.

In algorithmic modeling, each crow represents a potential solution to a problem, its flight space corresponds to the problem's solution space, and the food caching locations remembered by the crow correspond to the currently discovered optimal solutions. Suppose the population X has a size of N, and the position of crow i at the t-th iteration is represented by the vector xti, whose mathematical expression is given by equation (1), where Tmax represents the maximum number of iterations.

$$\boldsymbol{X} = \{\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \cdots, \boldsymbol{x}_N^t\}, i = 1, \cdots, N, t = 1, \cdots, T_{\max} \quad (1)$$

In algorithmic modeling, each crow represents a potential solution to a problem, with its position xti corresponding to a point in the solution space, and its stored food location mti corresponding to the currently discovered historical optimum. The population simulates crows' "tracking-countertracking" behavior through iterative updates. Its core location update strategy is divided into two scenarios based on the watchful AP of the stalked crow:

When the tracked crow is undetected (i.e., rj≥AP), the tracker crow i moves toward the food hiding position mtj of the tracked crow J. The position update formula is shown in equation (2), where fl is the flight step of the crow, controlling the search step; ri is a random number.

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + r_i \cdot fl \cdot \left(\boldsymbol{m}_j^t - \boldsymbol{x}_i^t\right) \quad (2)$$

When the tracked crow detects being followed (i.e., rj<AP), the tracked crow will fly to a random location to confuse the follower, and at this time the position of crow i is also updated to a random location. This mechanism aims to increase population diversity. The mathematical unified expression of the two position update strategies is shown in formula (3):

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{x}_i^t + r_i \cdot fl \cdot \left(\boldsymbol{m}_j^t - \boldsymbol{x}_i^t\right) & r_j \geq AP \\ \text{random} & \text{otherwise} \end{cases} \quad (3)$$

After each position update, crow i evaluates the fitness of the new position and updates the best position mti in its memory, as shown in equation (4):

$$m_i^{t+1} = \begin{cases} \boldsymbol{x}_i^{t+1} & f(\boldsymbol{x}_i^{t+1}) \leq f(\boldsymbol{m}_i^t) \\ \boldsymbol{m}_i^t & \text{otherwise} \end{cases} \quad (4)$$

Although the CSA principle is intuitive and its parameters are simple, its core mechanism has inherent defects. Its position update strategy lacks strong directionality, and fixed parameter settings are difficult to adapt to different stages of the search process, which directly leads to slow convergence and insufficient solution accuracy when dealing with complex problems.

3. Improved Crow Search Algorithm (ACSA)

3.1. ACSA Algorithm Framework

The ACSA algorithm continuously monitors the population diversity index dv, using this state variable that reflects the search process as a unified feedback signal, which is applied to guide individual selection and the adjustment of key parameters, thereby forming a dynamic cycle of perception, decision-making, and execution. This framework not only enhances the algorithm's adaptability to different problem characteristics but also improves its ability to escape local optima and conduct fine-grained searches.

3.1.1. Diversity-based adaptive guidance individual selection strategy

In the classic CSA algorithm, when crow j does not notice being followed by crow i , crow i updates its position by randomly selecting another crow j to follow. Although this mechanism can prevent the population from converging prematurely in a single direction and helps maintain population diversity, its randomized guidance pattern also causes the population's search direction to diverge and makes it difficult to focus, thereby hindering effective fine search in later stages and limiting the algorithm's convergence speed and solution accuracy. This study constructs multiple guiding individuals and proposes an adaptive selection strategy for guiding individuals based on the dynamic changes of population diversity during the evolutionary process.

To quantify population diversity, this paper uses Euclidean distance to measure differences between individuals and defines the diversity index as the ratio of the current generation's difference to that of the initial generation. Its calculation formula is shown in equation (5).

$$d_v = \frac{d^t}{d^0} \quad (5)$$

$$d^t = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i^t - \boldsymbol{x}_{best}^t\| \quad (6)$$

Here, dt represents the degree of difference in the population at generation t, d0 represents the degree of difference in the initial population, xtbest represents the optimal position of the population at generation t, and N represents the population size.

The position update mechanism of CSA is essentially a process of mutual learning among individuals. However, its single learning mode can easily lead to a rapid loss of population diversity, increasing the risk of the algorithm falling into local optima. To overcome this limitation, an adaptive guidance strategy based on dynamic perception of population diversity is proposed. This strategy adaptively switches between two population guidance modes by monitoring the level of population diversity in real time, in order to achieve an effective balance between exploration and exploitation capabilities. The specific process is as follows:

Excellent Group Model: Select food sources with fitness better than a certain threshold from the collected food sources to form an excellent group, and calculate their average position, as shown in Equation (7). In the early stages of evolution, the population diversity is relatively high. Using the average of the excellent group as guidance at this point can both prevent individuals from searching blindly and effectively delay the premature decay of diversity, which aligns with the algorithm's requirement to focus on broad exploration in the early stage. Here, a certain threshold is used to choose between the excellent group model and the random group model. Additionally, generating new search directions by averaging increases particle diversity and avoids the problem in PSO where particles merely follow a single global optimum (gbest), which makes them prone to getting trapped in local optima.

$$\mathrm{X}_{\text{group}} = \frac{1}{|S_p|}\sum_{j}^{|S_p|} m_j^t \quad (7)$$

Random group mode involves randomly selecting individuals to form a random group and calculating their average position, as shown in formula (8). When population diversity drops to a low level, using the random group guidance helps introduce perturbations, enhance population diversity, and thus assist the algorithm in escaping local optima. At the same time, this mode, by integrating global information, can guide the population to regroup in potentially high-quality regions.

$$\mathrm{X}_{\text{group}} = \frac{1}{|S_q|}\sum_{j}^{|S_q|} m_j^t \quad (8)$$

In summary, the proposed strategy establishes a dynamic

synergy mechanism between population diversity and guidance patterns. Population diversity, as an indicator during the search phase, is used to adaptively select the appropriate guidance method; different guidance patterns, in turn, regulate population diversity, forming a positive feedback loop: early in the evolution, the focus is on expanding the search range to enhance exploration; in the later stages of evolution, the focus shifts to improving convergence accuracy to optimize exploitation. This mechanism helps achieve a dynamic balance between global exploration and local exploitation, but its robustness in complex scenarios needs further validation. The unified expression for selecting guidance positions is shown in equation (9):

$$X_{\text{group}} = \begin{cases} \frac{1}{|S_p|}\sum_{j}^{|S_p|} m_j^t & d_v > \varepsilon \\ \frac{1}{|S_q|}\sum_{j}^{|S_q|} m_j^t & d_v \le \varepsilon \end{cases} \quad (9)$$

3.1.2. Adaptive Parameter Adjustment Strategy

The classic CSA uses a fixed flight length (fl) to control the search step size, without considering the dynamic needs of the search strategy during the iterative process. In the early stages of iteration, a too-small fl can limit global exploration capability, leading to local convergence caused by the decay of population diversity; while in the later stages of iteration, a too-large fl can cause the population to oscillate near the optimal solution, making fine-tuning difficult. To overcome the limitations of a fixed flight length, a dynamic feedback control system is established. The flight length fl changes from a static parameter to a dynamic parameter, whose value is adjusted according to the population state feedback. The fundamental motivation is that the algorithm's search behavior (controlled by fl) should respond to the results of the search (changes in population diversity). This is a state-feedback-based adaptive mapping control. When the population converges prematurely (possibly getting trapped in a local optimum), the strategy automatically increases the step size, applying a "disturbance" to force the algorithm to escape; when the population is effectively exploiting high-quality regions, the strategy reduces the step size to promote "convergence." This mechanism allows the algorithm to dynamically respond to the characteristics of different problems, providing stronger generality and robustness. The core advantage of this feedback mechanism is that it shifts the selection of the search strategy from pre-set to real-time decisions based on the actual optimization process, enabling the algorithm to adaptively balance exploration and exploitation, thereby significantly enhancing its capability to tackle complex optimization problems.

In summary, this study abandons the pre-set fixed variation pattern and instead constructs a real-time feedback-based adaptive mechanism. The core of this mechanism is the establishment of a dynamic "perception-decision" cycle: first, precisely perceive the current search state of the population; then, through a carefully designed mapping function, intelligently convert this state information into a specific search step size. This aims to transform fl from a static parameter that needs to be preset into a dynamic response variable driven by the algorithm's real-time performance, thereby achieving truly state-feedback-based adaptive mapping control.

Specifically, we use the Sigmoid function as the core of this mapping relationship, because it can provide smooth, nonlinear transition characteristics, effectively preventing abrupt parameter changes and enhancing the stability of the algorithm. This adaptive adjustment formula is defined as shown in equation (10):

$$\text{fl} = \text{fl}_{\min} + \frac{\text{fl}_{\max} - \text{fl}_{\min}}{1 + \exp(k \cdot (d_v - d_{\text{mid}}))} \quad (10)$$

Among them, flmax and flmin are the upper and lower limits of the flight length, constraining the basic range of fl; k is the slope parameter, which controls the steepness of the function curve; dmid is the function center point parameter. When dv is large (high population diversity), fl is close to flmin, and smaller step sizes are conducive to the algorithm performing fine-grained global exploration over a wide area. Conversely, when dv is small (low population diversity), fl is close to flmax, and larger step sizes help the population escape local optima. The parameter dmid can be regarded as the critical point for the transition between exploration and exploitation behaviors, and it is recommended to initially set it to 50% of the initial population diversity level (such as the average distance); the slope parameter k can be adjusted between 1 and 10 based on the problem dimension, with lower-dimensional or strongly perturbed problems tending toward larger k values (more sensitive response), and higher-dimensional or complex terrain problems tending toward smaller k values (more stable response).

3.1.3. Guided Individual Selection Strategy Based on Golden Sine

In the classic CSA algorithm, when crow j discovers that crow i is being followed, in order to protect its food from being stolen, it will move to a random location to mislead the follower. However, this random reset strategy has obvious drawbacks: first, blindly jumping in a vast solution space provides no guarantee of the quality of the new position, which can easily reduce search efficiency; second, in the later stages of algorithm development, when the population has already gathered near the global optimum, random jumps will disrupt convergence stability; moreover, this strategy fails to effectively utilize historical experience and global information, resulting in insufficient use of search information; finally, it lacks the ability to distinguish between search stages and cannot adapt to the differentiated needs for exploration and exploitation at different stages. Therefore, to address the above four shortcomings, this study introduces the golden ratio coefficient to improve the position update behavior of individuals in the 'being discovered' scenario. In the position update process of the Golden Sine Algorithm, the golden ratio coefficient is introduced to narrow the space, thereby guiding individuals to gradually approach the optimal value. The position update formula based on the golden ratio coefficient for individuals in the 'being discovered' scenario is shown in Equation (11).

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t \cdot |\sin(r_1)| - r_2 \cdot \sin(r_1) \cdot |s_1 \cdot \boldsymbol{x}_{best}^t - s_2 \cdot \boldsymbol{x}_i^t| \quad (11)$$

Among them, r1 and r2 are random numbers, achieving large-scale exploration through the periodic fluctuations of the sine function; xtbest is the global optimal position of the t-th generation of the population, ensuring the rationality of the search direction; s1 and s2 are golden ratio coefficients.

The core advantage of this improved strategy lies in that the generation of new positions always revolves around the current global optimal solution. The random perturbations

introduced by the sine function are directional and bounded, avoiding completely random disturbances, and significantly enhancing the targeting and efficiency of the search. During the development phase, even if an individual is "discovered," the executed "small-scale local perturbations" can ensure that the updated positions do not stray far from the current high-quality region, thereby maintaining the stability of the convergence process. Additionally, the strategy incorporates the golden ratio, which has been proven to be the optimal choice for efficient searching in uncertain intervals. This can systematically guide the contraction of the search space, and combined with the periodic characteristics of the sine function, enables diversified exploration near the global optimum.

3.2. Simulation Experiments and Result Analysis

3.2.1. Simulation Experimental Environment and Benchmark Functions

The programming language used in the experiment was Matlab (version R2021b), and it was run on a computer with the operating system Windows 10, an Intel(R) Core (TM) i5-7300HQ CPU with a clock speed of 2.50GHz, and 8.0GB of memory.

In this chapter, 10 basic test functions were selected for testing. These 10 basic test functions can be divided into unimodal functions and multimodal functions. The detailed information of these 10 basic test functions is shown in the figure 1 below.

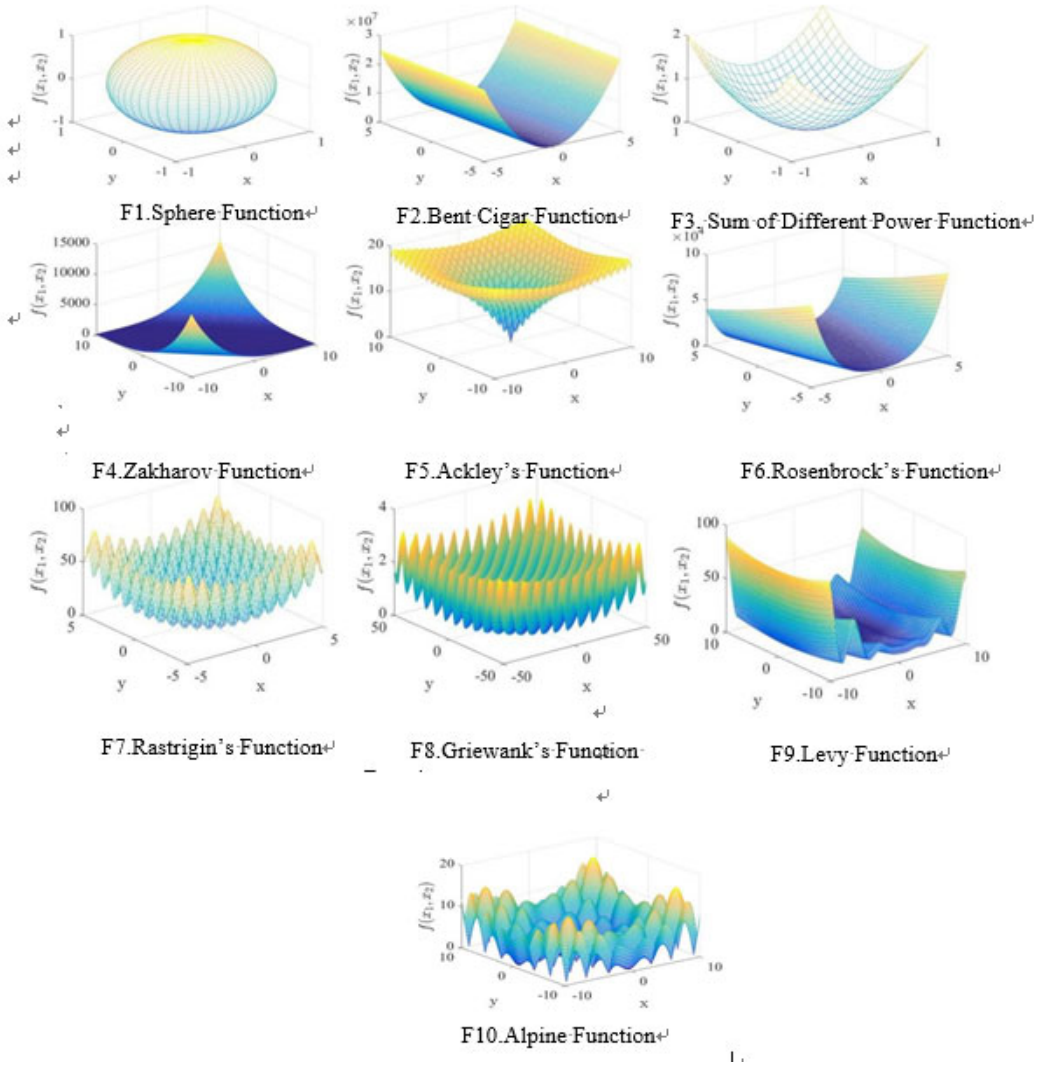

Fig.1 Images of Basic Test Functions

3.2.2. Experimental Parameter Settings

To systematically evaluate the performance of the ACSA algorithm, this study designed comparative experiments covering different dimensions and test functions. GWO, PSO, CSA, Multi-stage Disturbance Shared Crow Search Algorithm (MISCA), Sine-Cosine Guided Crow Search Algorithm (MCSA), and ACSA were selected to be tested on 10 basic test functions, and 30 independent experiments were conducted at dimensions of 10, 50, and 100. By comparing the means and standard deviations of each algorithm, the effectiveness of the improved algorithm's performance was verified.

Tab.1 Parameter Settings of Different Algorithms

Algorithm	Parameter Settings
GWO	None
PSO	C1=C2= 1.49618, w= 0.7298
CSA	AP=0.1, fl=2
MISCA	AP=0.1, fl=1,$\alpha_1 = 0.4, \alpha_2 = 0.7, \delta_1 = 0.4, \delta_2 = 0.001$
MCSA	AP=0.1, fl=2
ACSA	AP=0.1, flmax=2, flmin=0.5,$\varepsilon = 0.4, k = 5$

The parameters of the improved algorithm proposed in this

paper and those of the various algorithms used for comparative experiments are shown in Table 1. At the same time, to ensure the fairness of the experiments, the population size N for all algorithms in this paper is set to 30, and the maximum number of iterations Tmax is set to 1000. In addition, to avoid the influence of the initial population distribution on the comparison of algorithm performance, the experiments use 30 maps with different initial population distributions, and each algorithm runs once on each map, totaling 30 independent runs.

3.2.3. Experimental Results and Analysis

Tab.2 Experimental Results of Comparison Between Different Algorithms

Dimension	Functions	GWO		PSO		CSA		MISCSA		MCSA		ACSA	
		avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
10	F1	3E-117	8E-117	6.64725	15.9664	8.18471	3.84907	3.80107	4.03363	0.00027	0.00043	4E-166	0
	F2	3E-113	1E-112	9198.13	38406.4	47654.6	20838.7	14387.1	15123.1	3.05557	6.70973	1E-159	6E-159
	F3	1.1E-64	3.7E-64	8E+08	4.2E+09	905849	4601854	8.5E+08	4.2E+09	0.05143	0.05527	1E-219	0
	F4	1E-113	6E-113	0.37759	0.63669	0.23011	0.2405	0.15753	0.1844	5E-06	8.7E-06	7.9E-28	4.3E-27
	F5	4.5E-15	1.2E-15	4.5881	1.79114	2.65528	0.44275	2.16372	0.72777	0.00446	0.00512	8.9E-16	0
	F6	6.28008	0.71364	28.2702	37.0016	32.0225	28.2246	20.8898	22.7701	8.96844	0.01711	8.9302	0.03188
	F7	0.88059	2.12584	19.4268	8.88884	14.518	3.7341	18.5996	6.85958	0.00016	0.00032	0	0
	F8	0.01471	0.02213	0.6584	0.54209	0.97433	0.0929	0.76806	0.12426	0.0013	0.00288	0	0
	F9	0.0021	0.01152	0.9001	1.35256	0.02946	0.06555	0.40888	0.71854	0.20491	0.18499	0.70485	0.24998
	F10	6.1E-05	0.00015	1.12648	0.84221	0.23629	0.19609	0.69178	0.66733	0.00039	0.00035	7.5E-77	4.1E-76
50	F1	8.4E-44	1.3E-43	8300.46	2253.92	3284.88	725.896	2718.71	677.754	0.00168	0.0024	4E-168	0
	F2	1.3E-39	2.4E-39	8.7E+07	1.9E+07	3E+07	8402993	2.6E+07	8168179	31.70559	79.68932	2E-159	9E-159
	F3	5.3E-21	1.3E-20	3.2E+80	1.3E+81	3E+69	1.3E+70	1.3E+67	6E+67	0.21866	0.19395	3E-219	0
	F4	9.2E-43	3.4E-42	745.628	422.889	413.62	140.068	558.468	503.998	5.2E-05	0.00013	26.72	58.134
	F5	3.2E-14	3.5E-15	13.5532	1.10759	9.72361	0.80274	9.43746	0.8323	0.00629	0.00511	8.9E-16	0
	F6	47.3214	0.76923	41219.4	17271.3	8217.78	4029.96	5136.1	1970.75	48.98366	0.02211	48.71	0.05299
	F7	2.59653	6.17297	352.704	37.1069	324.281	26.495	293.599	28.9533	0.00062	0.00103	0	0
	F8	0.00352	0.00686	74.8206	17.3264	31.3375	9.16521	25.2778	6.23118	0.00583	0.01237	0	0
	F9	0.95359	0.3481	34.3482	12.582	11.441	3.92486	13.3244	7.3564	7.28904	1.60104	3.8947	0.59338
	F10	0.00013	0.00028	33.1114	7.30562	19.1045	2.81588	18.7955	4.15453	0.00193	0.00171	1.1E-85	5.9E-85
	F1	1.6E-29	2.4E-29	24687.4	5468.72	9260.48	1603.22	8036.44	1416.78	0.00304	0.004	3E-156	2E-155
	F2	2.3E-25	2.6E-25	2.4E+08	4.4E+07	8.7E+07	1.4E+07	7.9E+07	1.6E+07	52.0212	75.346	4E-155	2E-154
100	F3	1.2E-06	5.7E-06	4E+168	65535	2E+141	6E+141	6E+141	3E+142	0.3489	0.41233	4E-216	0
	F4	2.1E-29	1.9E-29	106092	341976	2044.57	911.759	285223	700671	0.0003	0.00077	221.45	190.56
	F5	1.1E-13	9.9E-15	14.7558	0.66772	11.0214	0.43917	10.4945	0.50962	0.00847	0.011	8.9E-16	0
	F6	97.5348	0.69422	191789	75032.6	25674.7	5333.18	24579.1	7800.97	98.9855	0.01541	98.407	0.09787
	F7	2.16951	4.03004	881.19	47.4533	832.687	34.1638	784.582	46.2179	0.0017	0.00358	0	0
	F8	0.00376	0.00812	235.062	53.0366	82.9126	12.0129	68.4492	9.87464	0.00531	0.01467	0	0
	F9	6.0303	1.43696	115.888	27.6462	39.8767	5.89906	36.3632	9.38369	22.202	2.63844	7.2913	0.87045
	F10	0.00026	0.00074	81.2126	9.31148	52.0176	4.3209	49.5524	5.95956	0.00304	0.00276	1.4E-81	7.4E-81

From the overall data in the table 2, the ACSA algorithm proposed in this paper demonstrates overwhelming performance advantages on the vast majority of test functions. Its optimization accuracy, convergence stability, and high-dimensional adaptability are all significantly superior to all comparison algorithms such as GWO, PSO, CSA, and MISCSA, achieving a breakthrough in comprehensive optimization performance by orders of magnitude. In terms of convergence accuracy, ACSA performs particularly well on unimodal functions (F1-F4) and multimodal functions (F5, F7, F8, F10). In full-dimensional scenarios of 10, 50, and 100 dimensions, the average accuracy of ACSA's optimization results on functions F1, F2, F3, F5, F7, F8, and F10 far surpasses that of other comparison algorithms by several

orders of magnitude. This fully demonstrates that through the improved search guidance strategy and adaptive parameter mechanism, ACSA simultaneously possesses strong local fine-tuning capabilities and efficient global optimal approximation ability, enabling it to precisely lock onto the theoretical optimal value ranges of the test functions.

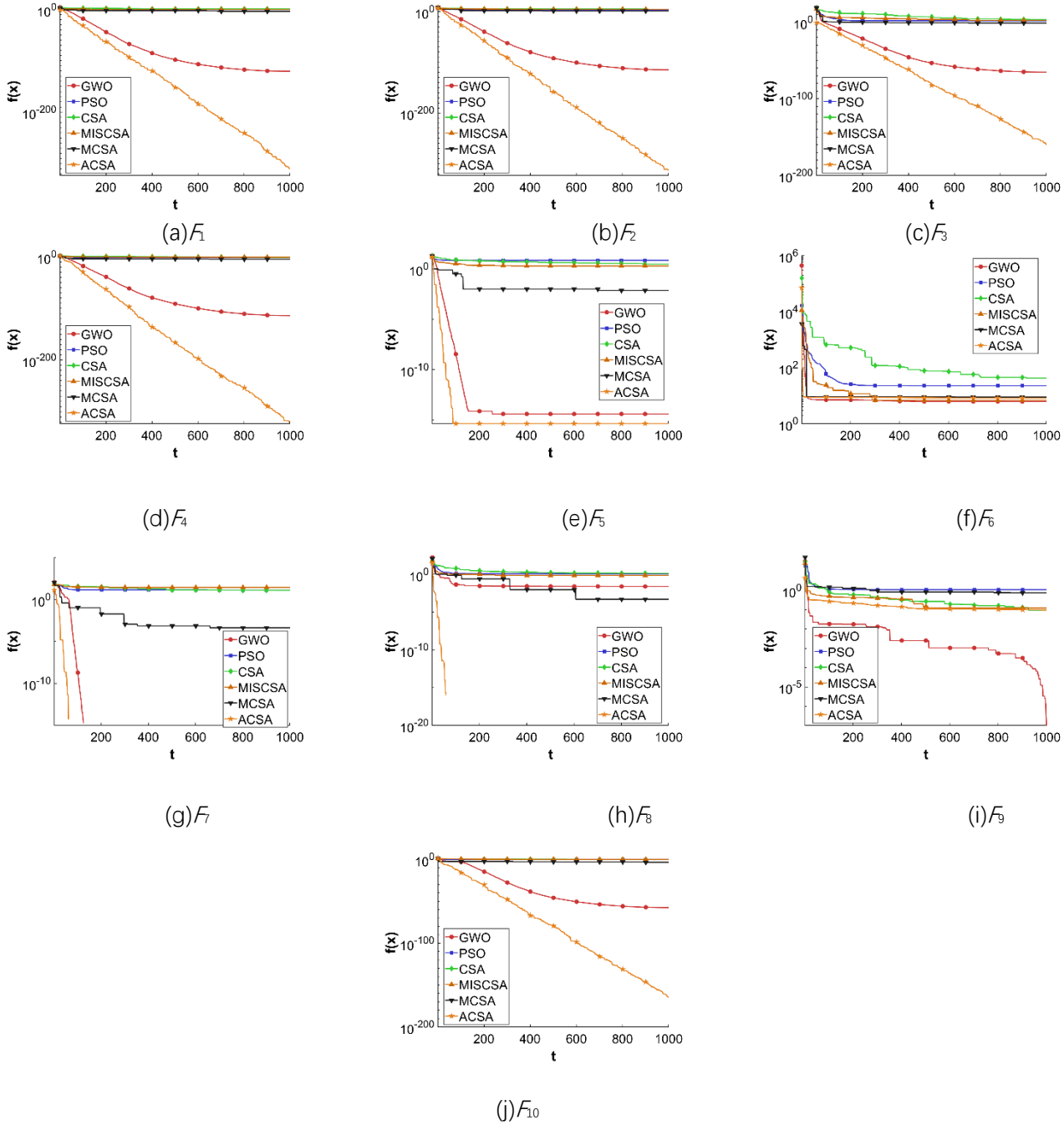

Fig.2 Convergence Curves of ACSA and Other Algorithms on Different Functions

To intuitively compare the dynamic convergence performance of various algorithms, the figure 2 above plots the convergence curves of ACSA and five comparison algorithms on 10 benchmark functions. A comprehensive analysis shows that the proposed ACSA demonstrates the best convergence characteristics in the vast majority of test cases.

First, in terms of convergence speed, the curve of ACSA (orange) generally has the steepest decline in the early iterations (especially noticeable for functions F4, F7, F8, etc.), indicating that its adaptive guidance strategy can efficiently steer the population toward promising search regions, with global exploration efficiency significantly superior to that of the comparison algorithms. Second, in terms of convergence accuracy, the ACSA curve eventually stabilizes at the lowest fitness value plateau (clearly visible in all subplots), confirming its powerful local exploitation capability, which can accurately locate high-quality solutions. Notably, when handling the complex multimodal function F10, ACSA continues to optimize after a rapid decline, whereas comparison algorithms such as PSO and CSA have long become trapped in local optima and stagnated, highlighting ACSA's advantage in balancing exploration and exploitation capabilities.

3.3. Summary

This paper addresses the core limitations of the classic

Crow Search Algorithm (CSA) in complex optimization scenarios, such as the imbalance between exploration and exploitation, rigid guidance strategies, fixed parameters, and weak boundary handling, and systematically proposes an improved Adaptive Crow Search Algorithm (ACSA). ACSA constructs a perception-decision-execution closed-loop optimization framework by introducing an adaptive guided individual selection strategy based on population diversity, a golden sine optimization mechanism, a memory bank update strategy, as well as dynamic parameter adjustment and reflective boundary handling.

Experimental results indicate that ACSA significantly outperforms comparative algorithms in terms of convergence accuracy, speed, and stability. As shown in the table data, ACSA can achieve solutions close to the theoretical optimum for unimodal functions and can effectively escape local optima for multimodal functions. The convergence curves further reveal that ACSA possesses both rapid global exploration and precise local exploitation capabilities; its curve has the steepest slope, the lowest plateau, and is smooth without oscillations. Dynamic analysis of population diversity confirms that ACSA can intelligently switch guidance modes according to real-time search states and maintain slight exploration in later stages to avoid premature convergence, demonstrating the superiority of its intrinsic adaptive mechanism.

The innovation of ACSA is not only reflected in performance improvement but also lies in providing a general optimization paradigm of 'state-aware–strategy-adaptive,' offering a methodological reference for the improvement of swarm intelligence algorithms.

References

[1] Talbi, E.-G. (2009). Book review: Metaheuristics: From design to implementation. European Journal of Operational Research, 205(2), 486. https://doi.org/10.1016/j.ejor.2009.12.039

[2] Glover, F. (1990). Tabu search: A tutorial. Interfaces, 20(4), 74-94. https://doi.org/10.1287/inte.20.4.74

[3] Storn, R., & Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11(4), 341-359. https://doi.org/10.1023/A:1008202821328

[4] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in Engineering Software, 69, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[5] Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures, 169, 1-12. https://doi.org/10.1016/j.compstruc.2016.03.001

[6] Hussien, A. G., Amin, M., Wang, M., Liang, G., Alshathri, S., & Elaziz, M. A. (2020). Crow search algorithm: Theory, recent advances, and applications. IEEE Access, 8, 173548-173565. https://doi.org/10.1109/ACCESS.2020.3024217

[7] Meraihi, Y., Gabis, A. B., Ramdane-Cherif, A., & Acheli, D. (2021). A comprehensive survey of crow search algorithm and its applications. Artificial Intelligence Review, 54(4), 2669-2716. https://doi.org/10.1007/s10462-020-09926-8

[8] Hamed Alnaish, Z. A., & Algamal, Z. Y. (2023). Improving binary crow search algorithm for feature selection. Journal of Intelligent Systems, 32(1), 20220228.

https://doi.org/10.1515/jisys-2022-0228

[9] Qu, C., & Fu, Y. (2019). Crow search algorithm based on neighborhood search of non-inferior solution set. IEEE Access, 7, 52871-52895.

https://doi.org/10.1109/ACCESS.2019.2908447

[10] Majhi, S. K., Sahoo, M., & Pradhan, R. (2020). A space transformational crow search algorithm for optimization problems. Evolutionary Intelligence, 13(3), 345-364. https://doi.org/10.1007/s12065-019-00259-8

[11] Han, X., Xu, Q., Yue, L., Dong, Y., & Zhang, Y. (2020). An improved crow search algorithm based on spiral search mechanism for solving numerical and engineering optimization problems. IEEE Access, 8, 92363-92382.

https://doi.org/10.1109/ACCESS.2020.2983985

[12] Khalilpourazari, S., & Pasandideh, S. H. R. (2020). Sine–cosine crow search algorithm: Theory and applications. Neural Computing and Applications, 32(11), 7725-7742. https://doi.org/10.1007/s00521-019-04543-9

[13] Lin, W.-T., & Li, C. (2024). Distributed asynchronous non-smooth optimization with coupled equality and bounded constraints. Neural Computing and Applications, 36(6), 2853-2866. https://doi.org/10.1007/s00521-023-08842-6

[14] Yuan, S., Wan, Y., Zhang, L., & Liu, Z. (2020). Control for hybrid systems: Applications and methods for adaptation and optimality. Optimal Control Applications and Methods, 41(6), 1811-1812. https://doi.org/10.1002/oca.2660