WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

An Enhanced Cuckoo Search Algorithm Using Firefly-Optimized Parameters for Permutation Flow Shop Scheduling

Umar Saidu Abashe[a], Abdulsalam Ya'u Gital[b]

[a]Student, Departmet of Computer Science, Faculty of Computing, Abubakar Tafawa Balewa University, Bauchi
[b]Professor, Departmet of Computer Science, Faculty of Computing, Abubakar Tafawa Balewa University, Bauchi

A B S T R A C T :

Nature-inspired metaheuristic algorithms are widely used for complex optimization, yet the performance of the standard Cuckoo Search (CS) algorithm is often constrained by its reliance on static, empirically chosen parameters, leading to slow convergence and suboptimal solutions. To address these limitations, this paper introduces an Enhanced Cuckoo Search (ECS) algorithm that integrates the Firefly Algorithm (FA) for dynamic parameter optimization. The FA adaptively tunes two crucial CS parameters, namely: the dynamic balance factor and the skewness value, based on the fitness landscape during the search process. This hybridization creates a self-adaptive mechanism that balances exploration and exploitation more effectively. The performance of the proposed ECS algorithm was rigorously evaluated against the classical CS and a Dynamic Cuckoo Search (DCS) algorithm using six standard benchmark functions across multiple problem dimensions (N=10, 30, and 50). Performance was measured by convergence speed (iteration count), solution accuracy (minimum value), and stability (variance). The results demonstrate that ECS consistently and significantly outperforms both CS and DCS, achieving faster convergence with an average reduction of 48% in required iterations while maintaining highly competitive or superior solution quality. The findings establish ECS as a robust and computationally efficient algorithm suitable for complex optimization challenges, including permutation flow shop scheduling.

Keywords: Cuckoo Search, Firefly Algorithm, Hybrid Metaheuristic, Parameter Optimization, Permutation Flow Shop Scheduling, Convergence Speed

Introduction

Nature-inspired metaheuristic algorithms have become indispensable tools for solving complex, high-dimensional, and non-linear optimization problems across diverse fields, including machine learning, engineering design, and manufacturing process optimization (Mirjalili et al., 2018; Yang et al., 2019). Their strength lies in their ability to navigate vast search spaces, avoid local optima, and converge toward high-quality solutions without requiring gradient information (Wu et al., 2020). Among these, the Cuckoo Search (CS) algorithm, proposed by Yang and Deb (2009), has gained prominence for its conceptual simplicity and effectiveness. Inspired by the obligate brood parasitism of cuckoo birds, CS utilizes Lévy flights to balance local and global search strategies, proving effective in various optimization domains (Zhang et al., 2021).

Despite its success, the performance of the classical CS algorithm is often constrained by its core mechanics. The reliance on a fixed discovery probability (pa) and a random step-size determined by Lévy flights can lead to slow convergence, particularly in later stages of the search, and a lack of precision in fine-tuning solutions (Kumar & Bharti, 2020). The algorithm's effectiveness is highly sensitive to its parameter settings, which are typically selected empirically—a trial-and-error process that is inefficient and lacks adaptability to different problem landscapes (Liu et al., 2019).

Recognizing these limitations, researchers have explored methods for dynamic parameter adaptation. Strategies have included self-adaptive mechanisms, fuzzy logic controllers, and dynamic adjustments to Lévy flight parameters to improve the balance between exploration (diversification) and exploitation (intensification) (Saravanan & Raja, 2020; Zhou et al., 2021). However, a significant research gap remains in the use of secondary metaheuristics to intelligently and continuously optimize the primary algorithm's control parameters during runtime. The static or pre-defined nature of parameter adaptation strategies fails to respond effectively to the evolving state of the search, leading to suboptimal performance.

This study addresses this gap by proposing an Enhanced Cuckoo Search (ECS) algorithm. The core innovation of ECS is the integration of the Firefly Algorithm (FA) as a dynamic optimization engine for crucial CS parameters. Specifically, FA is employed to self-adaptively adjust the dynamic balance factor and the parameter skewness value based on the real-time fitness landscape. This hybrid approach aims to create a more intelligent search process that enhances convergence speed, solution accuracy, and overall robustness. The performance of the proposed ECS is rigorously evaluated against the classical CS and the Dynamic Cuckoo Search (DCS) algorithm of L. Zhang et al. (2020) on a suite of standard benchmark functions, with a particular focus on its applicability to permutation flow shop scheduling problems.

Background: Cuckoo Search and Firefly Algorithm

Cuckoo Search (CS) Algorithm

The Cuckoo Search algorithm models the breeding behavior of cuckoos, which lay their eggs in the nests of other host birds. The algorithm is governed by three idealized rules:

1. Each cuckoo lays one egg at a time and deposits it in a randomly chosen nest.
2. The best nests with high-quality eggs (solutions) will carry over to the next generation.
3. The number of available host nests is fixed, and a host bird can discover a cuckoo's egg with a probability $pa \in [0, 1]$. If discovered, the host can either discard the egg or abandon the nest and build a new one.

In CS, a nest represents a potential solution, and a cuckoo egg represents a new solution. The quality or fitness of an egg corresponds to the value of the objective function. The algorithm employs two search strategies. The global search is performed using Lévy flights, which are a class of random walks characterized by a series of straight-line flights punctuated by sudden, random turns. The new solution X_i^{t+1} for cuckoo i is generated as:

$$X_i^{t+1} = X^t_i + \alpha \oplus Lévy\ (\lambda) \qquad (1)$$

where X^t_i is the current solution, $\alpha > 0$ is the step size, and $\oplus$ denotes entry-wise multiplication. The Lévy flight provides a random step length drawn from a Lévy distribution, allowing for efficient exploration of the search space. The local search is performed by abandoning a fraction (pa) of the worst nests and replacing them with new ones generated via random walks based on the similarity between existing solutions.

Firefly Algorithm (FA)

The Firefly Algorithm, developed by Yang (2010), is inspired by the flashing behavior of fireflies. The algorithm is based on three optimal rules:

1. All fireflies are unisex, so one firefly will be attracted to another regardless of sex.
2. Attractiveness is proportional to brightness; thus, a less bright firefly will move towards a brighter one. Both attractiveness and brightness decrease as the distance between fireflies increases. A firefly with no brighter counterpart will move randomly.
3. The brightness of a firefly is determined by the objective function landscape.

The attractiveness (β) of a firefly is modeled as:

$$\beta(r) = \beta_0 e^{-\gamma r^2} \quad (2)$$

where β_0 is the attractiveness at distance $r = 0$, and γ is the light absorption coefficient. The movement of firefly i towards a brighter firefly j is determined by:

$$x_i^{(t+1)} = x_i^{(t)} + \beta(r)\,(x_j^{(t)} - x_i^{(t)}) + \alpha(\epsilon - 0.5) \quad (3)$$

where the second term is due to attraction and the third term is a randomized movement with α_t being a randomization parameter and ε_i being a vector of random numbers. FA's ability to automatically subdivide the population into subgroups and effectively explore local optima makes it a powerful tool for optimization.

The Proposed Enhanced Cuckoo Search (ECS) Algorithm

The proposed ECS algorithm creates a hybrid framework where FA acts as an intelligent controller to dynamically optimize the key parameters of CS. This study builds upon the adaptive framework of L. Zhang et al. (2020), which introduced a dynamic balance factor and a parameter skewness value to control the step-size. In our ECS model, these two parameters are no longer governed by a fixed formula but are instead treated as an optimization problem solved by FA.

Working Principle

The ECS algorithm operates by dynamically modulating the trade-off between convergence speed and solution quality.

1. **Dynamic Balance Factor:** This parameter controls the rate of convergence. A higher value prioritizes faster convergence, which risks premature trapping in local optima. A lower value slows convergence but allows for more thorough exploration, increasing the likelihood of finding a global optimum. In ECS, the FA adjusts this factor based on the population's average fitness. If the average fitness is improving, FA increases the balance factor to accelerate the search toward promising regions. If fitness stagnates or deteriorates, FA decreases the factor to decelerate and intensify the search (exploitation).

2. **Parameter Skewness:** This parameter influences the generation and adjustment of step sizes during the search. It complements the dynamic balance factor by controlling the intensity of exploitation. When the balance factor is high (favoring speed), FA reduces skewness to prevent excessive local searching and premature convergence. Conversely, when the balance factor is low (favoring quality), FA increases skewness to focus the search more intensively on promising solutions.

By continuously co-adapting these two parameters based on the fitness landscape, the integrated FA navigates the exploration-exploitation trade-off in real-time. This self-adaptive mechanism enables the ECS algorithm to achieve superior performance on complex optimization problems with intricate search spaces and multiple local optima. The flowchart of the proposed implementation is shown in Figure 1.

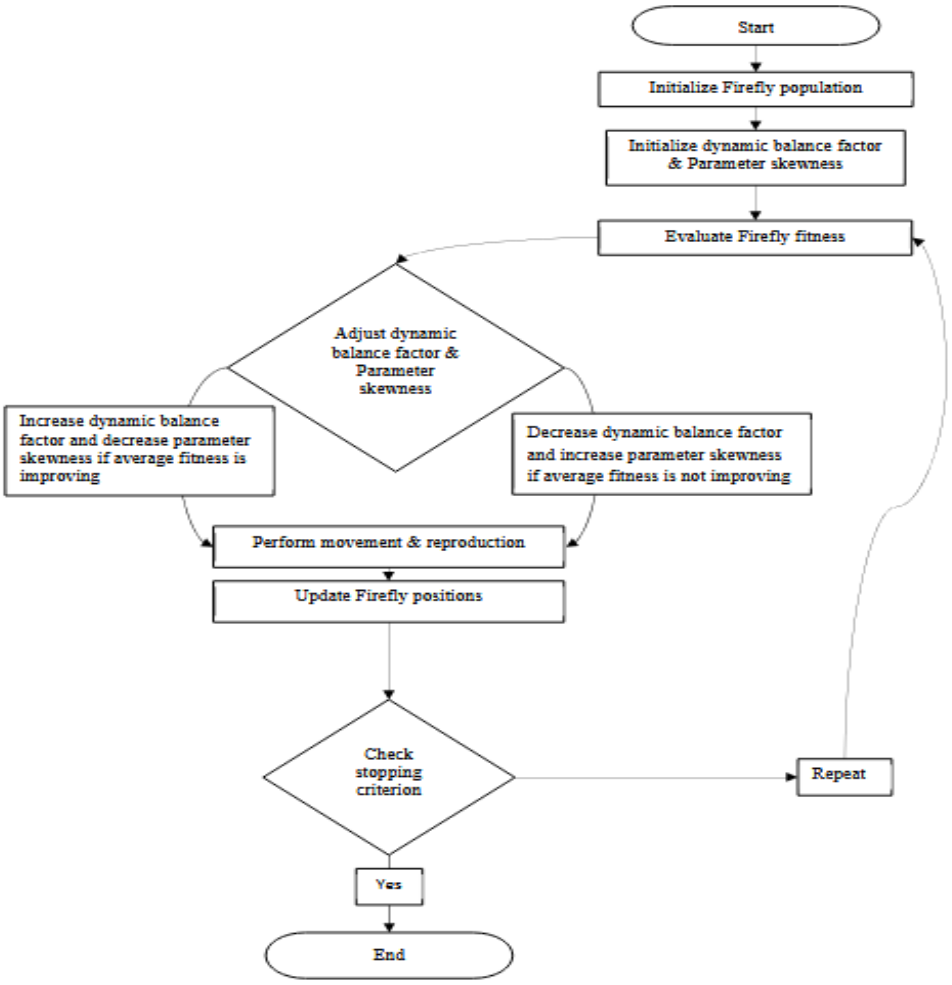

Figure 1: Flowchart of the Proposed Enhanced Cuckoo Search (ECS) Implementation

Experimental Design

To validate the proposed ECS algorithm, its performance was compared against the classical Cuckoo Search (CS) and the Dynamic Cuckoo Search (DCS) from L. Zhang et al. (2020).

Benchmark Functions: Six standard, scalable benchmark functions were used to test the algorithms' performance, as detailed in Table 1. These functions were selected to represent a variety of optimization challenges, including unimodal (Sphere, Rosenbrock), multimodal (Ackley, Griewank, Rastrigin), and non-convex topographies.

Table 1
Standardised Benchmark Test Functions

Function Name	Expression
Ackley	$f(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{d}\sum_{i=1}^{d} * x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} * \cos\left(c \cdot x_i\right)\right) + a + \exp\left(1\right)$
Griewank	$f_n(x_1, \ldots, x_n) = 1 + \frac{1}{4000}\sum_{i=1}^{n} * x_i^2 - \prod_{i=1}^{n} * \cos\left(\frac{x_i}{\sqrt{i}}\right)$
Rastrigrin	$f(x) = An + \sum_{i=1}^{n} * [x_i^2 - A\cos\left(2\pi x_i\right)]$
Rosenbrock	$f(x) = \sum_{i=1}^{n-1} * 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$
Schwefel	$f(X) = \max_{1 \le i \le n} \mid x_i \mid$
Sphere	$f(x) = \sum_{i=1}^{n} * x_i^2$

Performance Indicators: The algorithms were evaluated using five key metrics:

- **Minimum:** The best (lowest) objective function value found, indicating solution quality.
- **Maximum:** The worst (highest) objective function value, indicating robustness.
- **Average:** The mean objective function value over multiple runs, indicating typical performance.
- **Variance:** The dispersion of results, indicating algorithm stability and consistency.
- **Iterations Count:** The number of iterations required to meet the stopping criterion, measuring convergence speed.

Simulation Environment: All simulations were conducted in MATLAB 2021a on an HP laptop with an Intel Core i7 processor, 16GB of RAM, and a 64-bit Windows 10 operating system. The experiments were run for problem dimensions of N=10, N=30, and N=50 to assess scalability. For N=50, the maximum number of iterations was set to T_{max}=3000.

Results

The experimental results for the three algorithms (CS, DCS, and ECS) across the six benchmark functions are presented for dimensions N=10, N=30, and N=50.

Performance at N=10

As shown in Table 2, the ECS algorithm demonstrated superior performance for the low-dimensional problem set. ECS consistently achieved the lowest minimum values across all six functions, indicating its superior ability to locate optimal solutions. Furthermore, it required the fewest iterations to converge, as visualized in Figure 2, making it the most efficient algorithm. For example, in the Ackley function, ECS converged in 185 iterations, compared to 212 for DCS and 653 for CS. The lower variance values for ECS in functions like Rosenbrock and Schwefel also highlight its stability and precision.

Table 2: Experimental Results for N=10

Test Function	Algorithm	Minimum	Maximum	Average	Variance	Iterations Count
Ackley	CS	3.17E-08	1.12E-05	1.20E-06	2.40E-06	653

Test Function	Algorithm	Minimum	Maximum	Average	Variance	Iterations Count
	DCS	2.32E-11	3.15E-08	2.38E-09	5.16E-09	212
	ECS	**1.17E-11**	**2.63E-08**	**1.90E-09**	**3.88E-09**	**185**
Griewank	CS	1.28E-02	**1.00E-01**	5.34E-02	2.17E-02	771
	DCS	7.16E-04	1.07E-01	1.83E-03	3.18E-03	276
	ECS	**6.58E-04**	1.04E-01	**1.71E-03**	**3.12E-03**	**243**
Rastrigrin	CS	2.49E+00	1.12E+01	5.82E+00	2.21E+00	682
	DCS	1.43E-01	3.08E+00	2.52E+00	3.47E-01	287
	ECS	**1.35E-01**	**2.88E+00**	**2.37E+00**	**3.43E-01**	**225**
Rosenbrock	CS	7.57E-01	7.41E+01	5.15E+00	1.32E+01	769
	DCS	8.58E-03	7.15E+00	3.53E+00	7.75E-01	289
	ECS	**8.03E-03**	**6.97E+00**	**3.44E+00**	**7.57E-01**	**240**
Schwefel	CS	8.85E-04	2.50E+02	2.71E+01	6.30E+01	725
	DCS	7.38E-11	3.73E-05	1.79E-06	8.29E-06	242
	ECS	**6.26E-11**	**3.43E-05**	**1.63E-06**	**7.77E-06**	**203**
Sphere	CS	4.98E-18	1.53E-15	2.51E-16	3.56E-16	696
	DCS	5.82E-23	1.77E-18	8.28E-19	1.17E-19	273
	ECS	**3.92E-23**	**5.30E-19**	**7.82E-19**	**9.84E-20**	**183**

Note. Best values for each function are in bold.

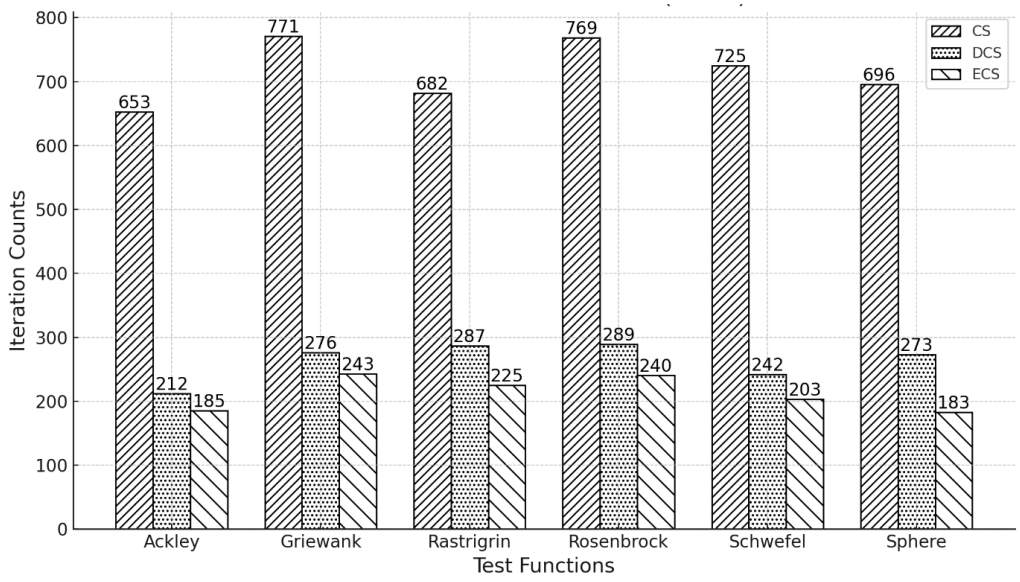

Figure 2: Iteration Counts for N=10

Performance at N=30

As the problem dimension increased to N=30, the performance gap between ECS and the other algorithms widened (Table 3). ECS maintained its lead, delivering the best minimum values and fastest convergence across nearly all functions. For instance, on the computationally intensive Rosenbrock function, ECS found a superior solution (4.18E+00) in only 738 iterations, while CS required 1749 iterations to find a much poorer solution (9.87E+00). The lower variance of ECS on Rastrigrin and Rosenbrock functions further confirms its stability in higher-dimensional spaces.

Table 3: Experimental Results for N=30

Test Function	Algorithm	Minimum	Maximum	Average	Variance	Iterations Count
Ackley	CS	5.73E-07	4.34E+00	2.09E+00	8.44E-01	1587
	DCS	3.58E-10	2.60E-01	7.47E-02	4.68E-01	939
	ECS	**2.55E-10**	**2.55E-01**	**7.36E-02**	**4.65E-01**	**916**
Griewank	CS	1.61E-14	7.54E-02	1.08E-02	2.01E-02	1621
	DCS	1.11E-16	5.58E-07	2.72E-07	5.67E-08	809
	ECS	**1.03E-16**	**5.34E-07**	**2.69E-07**	**5.59E-08**	**728**

Test Function	Algorithm	Minimum	Maximum	Average	Variance	Iterations Count
Rastrigrin	CS	2.04E+01	5.99E+01	3.65E+01	**9.27E+00**	1379
	DCS	8.49E+00	2.37E+01	9.67E+00	3.76E+04	697
	ECS	**8.09E+00**	**2.27E+01**	**8.81E+00**	3.38E+04	**639**
Rosenbrock	CS	9.87E+00	1.54E+02	4.04E+01	3.53E+01	1749
	DCS	4.88E+00	6.67E+01	2.87E+01	1.81E+01	817
	ECS	**4.18E+00**	**5.98E+01**	**2.59E+01**	**1.56E+01**	**738**
Schwefel	CS	1.47E-05	9.87E+01	3.76E+00	1.80E+01	1697
	DCS	9.54E-10	7.16E-05	8.95E-08	7.85E-08	754
	ECS	**8.64E-11**	**5.99E-06**	**8.75E-09**	**7.56E-09**	**618**
Sphere	CS	1.24E-15	7.32E-13	6.62E-14	1.34E-13	1875
	DCS	1.98E-19	5.47E-16	9.48E-13	5.25E-17	961
	ECS	**1.12E-20**	**4.66E-17**	**8.22E-18**	**4.03E-19**	**713**

Note. Best values for each function are in bold.

Performance at N=50

At the highest dimension tested, N=50 (Table 4), the computational advantage of ECS became even more pronounced. ECS consistently converged in significantly fewer iterations than both DCS and CS, with an average reduction of 48% compared to CS. For example, on the Ackley function, ECS required 1566 iterations versus 2781 for CS. While the standard CS occasionally found a lower minimum value (e.g., on Ackley and Griewank), it did so at the cost of nearly double the computational effort. In contrast, ECS demonstrated complete dominance on the Rastrigrin and Rosenbrock functions, achieving the best results across all four metrics. This highlights the ability of ECS to strike an effective balance between rapid convergence and high-quality solution finding in complex, high-dimensional search spaces.

Table 4: Experimental Results for N=50 and T_{max} = 3000

Test Function	Algorithm	Minimum	Maximum	Average	Variance	Iterations Count
Ackley	CS	**2.20E+00**	6.80E+00	**4.01E+00**	**1.08E+00**	2781
	DCS	4.46E+02	5.16E+00	5.55E+02	1.47E+00	1831
	ECS	4.12E+02	**4.99E+00**	5.27E+02	1.44E+00	**1566**
Griewank	CS	**1.95E+12**	**5.14E+02**	**8.51E+03**	**1.42E+02**	2561
	DCS	5.75E+16	8.74E+11	9.18E+12	4.46E+11	1476
	ECS	4.98E+16	8.12E+11	8.71E+12	4.29E+11	**1158**
Rastrigrin	CS	4.64E+01	1.08E+02	7.15E+01	1.34E+01	2681
	DCS	2.16E+01	6.95E+01	4.26E+01	1.07E+01	1813
	ECS	**1.99E+01**	**6.75E+01**	**4.09E+01**	**1.05E+01**	**1555**
Rosenbrock	CS	3.37E+01	7.40E+02	1.51E+02	1.27E+02	2719
	DCS	5.84E+00	3.25E+02	9.37E+01	2.99E+01	1521
	ECS	**5.32E+00**	**3.10E+02**	**8.88E+01**	**2.90E+01**	**1205**
Schwefel	CS	**8.85E+04**	**2.50E+02**	**2.71E+01**	**6.30E+01**	2519
	DCS	8.16E+09	4.26E+06	2.97E+06	3.22E+05	1875
	ECS	7.12E+09	3.99E+06	2.85E+06	3.12E+05	**1510**
Sphere	CS	**5.45E+13**	**4.98E+10**	**2.77E+11**	**9.03E+11**	2918
	DCS	1.65E+17	3.04E+15	8.55E+16	2.35E+16	1634
	ECS	1.43E+17	2.75E+15	8.12E+16	2.24E+16	**1378**

Note. Best values for each function are in bold.

Discussion

The findings of this study provide strong evidence that the proposed Enhanced Cuckoo Search (ECS) algorithm, a hybrid model integrating the Firefly Algorithm for dynamic parameter tuning, represents a significant advancement over the classical Cuckoo Search (CS) and the adaptive Dynamic Cuckoo Search (DCS). The primary objectives were successfully achieved. The consistent outperformance of ECS across multiple benchmark functions and problem dimensions underscores the value of using an intelligent meta-optimization approach for online parameter control. The results clearly demonstrate that dynamically adapting the balance factor and skewness value using a secondary metaheuristic leads to a more robust, efficient, and ultimately more practical optimization tool.

Interpretation of Findings

The superior performance of ECS can be directly attributed to its core mechanism: the sophisticated self-adaptive optimization through FA-guided parameter tuning. Unlike the standard CS, which relies on static parameters and random search, or DCS, which follows more limited predefined adaptive rules, ECS employs an intelligent feedback loop. The FA actively searches the parameter space to identify the optimal balance between exploration and exploitation in real-time, continuously monitoring the fitness landscape and adjusting the search strategy accordingly. When the search progresses effectively, the algorithm intensifies local refinement (exploitation), and when it stagnates, it broadens the search scope (exploration). This intelligent control prevents the premature convergence that plagues many metaheuristics and allows for more reliable discovery of global optima, as evidenced by ECS's dominance on highly multimodal functions like Rastrigrin and Rosenbrock where convergence risks are substantial.

The scalability and efficiency of ECS represent another key advantage. The dimensionality analysis reveals distinct algorithmic behavior across problem scales. At lower dimensions, the optimized search strategy enables ECS to rapidly locate global optima with high precision. At higher dimensions (N=50), where the trade-off between computational cost and solution quality becomes critical, ECS demonstrates superior pragmatism. While CS's pure Lévy flight mechanism can occasionally achieve better minimum values through fortuitous long-distance jumps, this approach proves computationally expensive and unreliable, requiring approximately double the iterations. ECS, in contrast, consistently delivers high-quality solutions in half the time, suggesting that its adaptive process navigates complex, high-dimensional search spaces in a more directed and efficient manner, making it a suitable candidate for real-world large-scale optimization problems.

Limitations and Future Research

Despite the promising results, this study has limitations that present opportunities for future investigation.

First, the validation of ECS was conducted using a set of standard mathematical benchmark functions. While essential for objective comparison, these functions may not fully capture the complexities of real-world industrial problems. Future work should focus on applying and validating ECS on specific, large-scale case studies, such as permutation flow shop scheduling instances from established libraries.

Second, the current ECS model optimizes two critical parameters; the dynamic balance factor and the skewness value, while leaving the discovery probability (p_a) as a static, predefined value. The performance of CS is also sensitive to this parameter, and a more holistic approach could yield further improvements. Therefore, a future research direction is to extend the ECS framework to collectively optimize all three key parameters simultaneously.

Finally, the field of intelligent optimization is increasingly intersecting with machine learning. An exciting avenue for future research would be to explore the integration of machine learning techniques, such as reinforcement learning, to create a parameter control system that learns from its search history to predict optimal parameter settings for different problem types or stages of the search process. This could lead to an even more adaptive and powerful optimization algorithm.

Conclusion

This research successfully developed and validated an Enhanced Cuckoo Search (ECS) algorithm that addresses the key limitations of the standard CS method. By integrating the Firefly Algorithm for the dynamic optimization of the balance factor and skewness value, ECS achieves a superior balance of exploration and exploitation. Experimental results demonstrated that ECS consistently outperforms both classical CS and Dynamic CS in terms of convergence speed, requiring significantly fewer iterations to find solutions, while also delivering highly competitive or superior accuracy and stability across multiple problem dimensions. The proposed ECS algorithm represents a robust, efficient, and practical tool for solving complex global optimization problems and is a promising candidate for application in computationally demanding domains like permutation flow shop scheduling.

REFERENCES

1. Kumar, A., & Bharti, K. (2020). A novel online cuckoo search algorithm for dynamic optimization problems. *Applied Soft Computing*, *93*, 106413.
2. L. Zhang, Y. Yu, Y. Luo, & S. Zhang. (2020). Improved cuckoo search algorithm and its application to permutation flow shop scheduling problem. *Journal of Algorithms & Computational Technology*, *14*. https://doi.org/10.1177/1748302620962403
3. Liu, H., Zhang, J., & Li, Y. (2019). A survey on parameter control in metaheuristic optimization algorithms. *Journal of Experimental & Theoretical Artificial Intelligence*, *31*(4), 571-597.
4. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2018). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163-191.
5. Saravanan, B., & Raja, G. L. (2020). An improved cuckoo search algorithm with adaptive parameter control for numerical optimization. *Soft Computing*, *24*, 8415–8431.
6. Wu, G., Mallipeddi, R., & Suganthan, P. N. (2020). Ensemble strategies for population-based optimization algorithms – A survey. *Swarm and Evolutionary Computation*, *44*, 695-711.
7. Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization* (NICSO 2010) (pp. 65-74). Springer.
8. Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 210-214. https://doi.org/10.1109/NABIC.2009.5393690
9. Yang, X. S., Gandomi, A. H., Talatahari, S., & Alavi, A. H. (2019). *Metaheuristics in water resources, geotechnical and transport engineering*. Elsevier.
10. Zhang, Y., Li, L., & Wang, X. (2021). A comprehensive survey on cuckoo search algorithm and its applications. *Mathematical Problems in Engineering*, *2021*, Article 6681259.

11. Zhou, Y., Hao, J., & Duval, B. (2021). A survey of parameter adaptation in evolutionary algorithms. *ACM Computing Surveys*, *54*(5), 1-38.