STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **16**, December 2026, pp 6221–6258.
Published online in International Academic Press (www.IAPress.org)

A Multi-Chaotic Enhanced Human Evolutionary Optimization Algorithm with FPGA Acceleration for High-Performance Global Optimization

Rabab Ouchker [1], Ismail Mchichou [1], Ahmed Bencherqui [2,*], Mohamed Amine Tahiri [2], Hicham Amakdouf [1], Mhamed Sayyouri [2]

[1] *Laboratory of Electronic Signals and Systems of Information, Sidi Mohamed Ben Abdellah University, Fez, Morocco*
[2] *Laboratory of Engineering, Systems and Applications, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco*

Abstract Metaheuristic optimization algorithms frequently suffer from premature convergence towards local optima due to inadequate exploration diversity and fixed search strategies. This paper presents MC-HEOA (Multi-Chaotic Enhanced Human Evolutionary Optimization Algorithm), which combines three contributions: (1) the integration of four complementary chaotic maps (Logistic, Tent, Sine, Henon) selected according to dimensional diversity, ergodic properties, and differentiated mixing speeds; (2) a dynamic voting system that selects the operative chaotic map at runtime from the fitness observed at the current iteration, rather than from a choice fixed in advance; and (3) an FPGA implementation of the resulting algorithm on an Artix-7 XC7A100T device. Unlike existing static chaotic integration methods that rely on a predefined map, the proposed mechanism defers the choice of chaotic dynamics to execution, without manual configuration or problem-specific tuning. Experimental validation covers twelve classical benchmark functions, the CEC2022 suite in ten dimensions, and two structural engineering problems (Welded Beam Design and Tension/Compression Spring). The results show that the map which prevails depends on the problem considered: the Logistic map is retained in more than 89% of iterations on the engineering problems, whereas the Henon variant obtains the best average rank on CEC2022, which supports runtime selection over a fixed choice. Against seven recent metaheuristics on CEC2022, the proposed variants outperform WOA, COA and C-KOA across the twelve functions and lead on the hybrid functions F6 and F11, while remaining behind TT-BWKO in overall rank; a Friedman test confirms the significance of the observed ordering ($p = 3.45 \times 10^{-10}$). On the Welded Beam problem, MC-HEOA reaches a cost of 1.6742, improving on the best previously reported result by 2.97%, while on the Tension/Compression Spring problem it is competitive without establishing a new best. The hardware implementation operates at 200 MHz within 2.3 W and occupies 79.6% of the available logic, which positions the algorithm for time-critical optimization in energy-constrained embedded settings.

Keywords Chaotic optimization; HEOA; FPGA implementation; dynamic voting system; metaheuristic algorithms

AMS 2010 subject classifications 90C59, 68W50

DOI: 10.19139/soic-2310-5070-3946

1. Introduction

Engineering design optimization problems consist of selecting a set of parameters (variables) to achieve the optimal value of the design index (objective) under a series of relevant constraints. These challenges exist widely in various domains, such as production cost optimization, public transportation optimization, production decision optimization, structural design optimization, feature selection, trajectory planning, and many other contemporary engineering applications [1, 2, 3, 4].

*Correspondence to: Ahmed Bencherqui (Email: ahmed.bencherqui@usmba.ac.ma). Laboratory of Engineering, Systems and Applications, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)
Copyright © 2026 International Academic Press

To effectively solve these complex optimization problems, metaheuristic algorithms (MHAs) constitute a privileged approach. Their central principle consists of progressively approaching the optimal solution through iterative search in the solution space. Unlike classical optimization methods such as gradient descent or Newton-Raphson which require precise gradient information and convexity assumptions, MHAs present decisive advantages: (1) derivative-free operation applicable to non-differentiable and black-box functions, (2) global search capability with local minima escape mechanisms, (3) minimal assumptions allowing treatment of discrete and constrained problems, (4) inherent robustness to noise and parametric uncertainties, and (5) parallelization potential on modern hardware architectures [5, 45]. The continuous evolution of computational capabilities and hardware acceleration techniques such as FPGA implementation strengthen their applicability to real-time optimization scenarios.

The diversity of search mechanisms has led to a classification of MHAs into four main categories, as illustrated in Figure 1: swarm algorithms, evolutionary algorithms, nature-inspired algorithms, and mathematical-type algorithms.

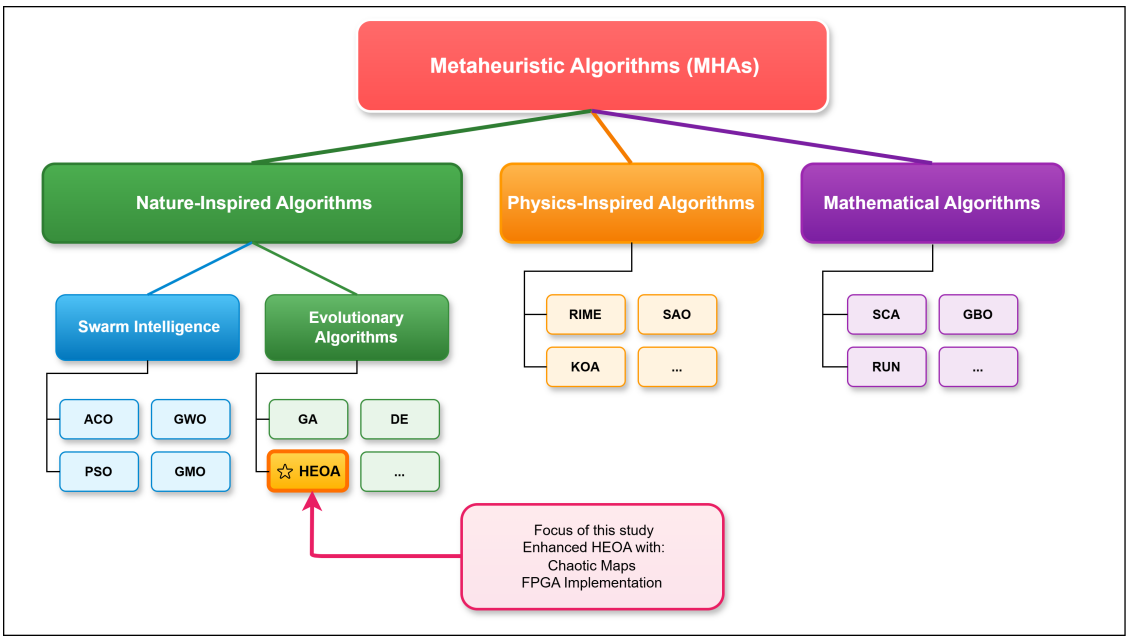


Figure 1. Hierarchical classification of metaheuristic optimization algorithms

For swarm algorithms, they currently constitute the most popular MHAs, inspired by adaptive and self-organized group behaviors of social species in nature, such as foraging, hunting, working, or migration [6, 7, 8]. For example, in the classical ant colony optimization algorithm (ACO), the foraging behavior based on pheromone concentration in the ant group is mathematically modeled to form a basic search framework. The grey wolf optimizer (GWO) was constructed by simulating the hunting behavior of grey wolves, belonging to swarm algorithms. Based on the hierarchical mechanism and hunting behavior, this special mechanism achieved good results in balancing exploration and exploitation of GWO, thus ensuring an effective balance in convergence speed and solution accuracy.Similar swarm-based approaches have been applied to energy-constrained settings, such as pathfinding in wireless sensor networks for the Internet of Things [67].

Evolutionary algorithms are built according to the natural law of survival of the fittest. For the genetic algorithm (GA), it simulates Darwinian biological evolution theory, mainly including natural selection and genetic mechanisms. The differential evolution algorithm (DE) was also a representative of this type, inspired by mutation strategy, crossover strategy, and selection operations. Among evolutionary algorithms, the Human Evolutionary Optimization Algorithm (HEOA) distinguishes itself through its biomimetic approach to the human evolution process. Recently proposed in 2024, HEOA has demonstrated competitive performance on various test functions

thanks to its structure. This algorithm, organised around two distinct phases, human exploration and human development, aims to balance the global exploration and local exploitation capabilities of the optimization process. However, like most metaheuristic algorithms, HEOA can benefit from targeted improvements to achieve faster convergence and better solution quality. Nature-inspired algorithms are developed by modeling common natural phenomena, such as rain, snow, wind, etc. Mathematical-type algorithms mainly include the sine cosine algorithm (SCA), gradient-based optimizer (GBO), and Runge Kutta optimizer (RUN) [9, 10].

Although optimization problems can be solved by various MHAs, accuracy and efficiency differ due to diverse algorithmic structures. However, despite the promising performance of MHAs, premature convergence towards local optima remains a major challenge that significantly limits their performance in solving complex optimization problems. While metaheuristic algorithms cannot mathematically guarantee global optimum convergence due to their stochastic nature, this limitation necessitates the development of targeted improvement strategies to reduce the risk of premature convergence, enhance exploration capabilities and maintain population diversity during the optimization process [11, 12, 13]. The convergence behaviour of swarm-based algorithms has itself been the object of dedicated analysis, which relates algorithmic mechanisms to the structure of the problem being solved [66].

To improve the performance of original algorithms, different enhanced versions of basic MHAs are introduced [14, 15]. Two common types of improvement methods are discussed here. In the first class, hybrid algorithms are constructed by mixing different basic algorithms with various characteristics. However, this type of method may inherit their own deficiencies when combining the advantages of different algorithms, which means certain limitations on improvement efficiency. Thus, another type of method is studied, that is, multiple improvement strategies are incorporated into the original algorithms. For example, aiming to overcome the defects of low accuracy, slow convergence speed, and trapping in local optima, targeted improvement strategies are introduced [16].

The integration of chaotic maps into metaheuristic optimization algorithms represents a promising approach to improve exploration capabilities and reduce the risk of trapping in local optima [17, 18, 19]. Chaotic systems possess properties such as sensitivity to initial conditions, ergodicity, and non-periodicity, which can enrich population diversity and enhance the balance between exploration and exploitation. Chaotic maps bring specific dynamic properties to the optimization process, generating pseudo-random sequences that support the exploration of complex optimization landscapes. Several recent works have demonstrated the effectiveness of chaotic injection in various optimization algorithms, using maps such as Logistic, Tent, Henon, and Sine to generate diversified dynamic behaviors.

The hardware implementation of metaheuristic algorithms on Field-Programmable Gate Array (FPGA) constitutes another important research direction to achieve computational acceleration in real-time applications. FPGAs offer advantages in terms of parallelism, reconfigurability, and energy efficiency compared to conventional processors. The intrinsic parallel architecture of FPGAs enables simultaneous execution of multiple optimization operations, reducing computation time for complex optimization problems while preserving numerical precision, making the approach suitable for time-critical optimization scenarios in engineering applications [20, 21, 22, 23].

It is fundamental to recognize that metaheuristic algorithms, due to their stochastic nature and random initialization, cannot mathematically guarantee convergence to the global optimum or exhaustive exploration of the search space. Unlike deterministic methods such as adaptive observers with Lyapunov stability proofs establishing fixed-time convergence [48], or neural network-based approaches with theoretical guarantees [49, 50, 51, 52, 53, 54], metaheuristics constitute practical approaches aimed at statistically improving solution quality rather than formally proving global convergence. This distinction positions metaheuristics as efficient solutions for problems where deterministic methods are inapplicable (non-differentiable functions, discontinuous spaces, high dimension), while acknowledging the impossibility of absolute theoretical guarantees. In this context, MC-HEOA is presented as an improvement of HEOA aimed at reducing the risk of premature convergence through adaptive integration of complementary chaotic dynamics, without claiming to eliminate the inherent limitations of stochastic approaches.

In this context, this paper presents an enhanced version of HEOA that exploits the dynamic properties of chaotic maps to address these limitations. The main contributions of this work are:

(1) An MC-HEOA algorithm integrating four distinct chaotic maps (Logistic, Tent, Henon, and Sine) incorporated into the exploration and exploitation phases to improve population diversity and reduce the risk of trapping in local optima.

(2) A dynamic voting system that selects the operative chaotic map at runtime from the fitness observed at the current iteration, rather than from a choice fixed in advance, thereby removing the need for a preliminary experimentation phase to identify a suitable map.

(3) An FPGA implementation of the algorithm on an Artix-7 XC7A100T device, operating at 200 MHz within a 2.3 W power envelope, with a documented Q4.8 fixed-point representation and a quantified resource budget.

(4) An experimental validation covering twelve classical benchmark functions, the CEC2022 suite in ten dimensions, and a comparison with seven recent metaheuristics from the literature, supported by a Friedman rank test.

(5) Validation on two structural engineering optimization problems (Welded Beam Design Problem and Tension/Compression Spring Problem), establishing a new best reported cost on the former.

The remainder of this paper is organised as follows. Section 2 reviews chaotic integration approaches in metaheuristic algorithms and positions the present contribution with respect to recent multi-strategy advances. Section 3 presents the original HEOA algorithm and the four chaotic maps employed. Section 4 details the construction of MC-HEOA, the selection mechanism and the FPGA architecture. Section 5 reports the experimental results, covering hardware validation, benchmark performance, the comparison with the state of the art, and the two engineering problems. Section 6 discusses the limitations of the present work and outlines research perspectives, and Section 7 concludes.

2. State of the Art

The use of metaheuristic algorithms to solve complex problems in engineering, medical image processing, and applied mathematics has emerged as an essential approach. However, these methods still face significant challenges such as premature convergence, ineffective exploration of the optimization landscape, and imbalance between exploration and exploitation phases. Faced with these limitations, numerous studies have been proposed to overcome these defects by incorporating chaotic maps into the search procedure. This approach replaces classical pseudo-random generators with chaotic sequences, which increases the diversity of solutions and the ability to avoid local optima.

2.1. Chaotic Integration Approaches in Metaheuristic Algorithms

The Modified Black-Winged Kite Optimizer (M-BWKO) algorithm [24] presents a particularly innovative method in this context, based on six major improvements. Among these, one of the most significant is chaotic reactivation conditioned by population diversity. The M-BWKO also incorporates chaotic weighting and adaptive Cauchy adjustment. In a similar approach, the Chaotic Archimedean Optimizer (CAO) [25] proposes to incorporate ten different chaotic maps including notably Logistic, Tent, Henon and Sine, similar to those exploited in our MC-HEOA approach, in the three main components of the initial algorithm: initialization, volume and density update, and position update. Although this method is based on static map selection, it considerably improves the balance between exploration and exploitation.

Following this same approach, the Chaotic Kepler Optimization Algorithm (CKOA) [26] improves the algorithm based on Kepler's laws by incorporating a chaotic dynamic diversification strategy. Although the algorithm is not explicitly focused on adaptive control of chaotic maps, it stands out through better performance on CEC benchmark tests and engineering problems, with rigorous evaluation through Wilcoxon tests and convergence curves. Similarly, the Chaotic White Shark Optimizer (CWSO) [27] relies on the WSO algorithm structure while adding ten chaotic maps, implemented statically in key elements of the process. The algorithm shows significant improvement in terms of solution quality, convergence speed, and resilience against local optima. Nevertheless, as with CAO, the selection of maps is not dynamic but fixed in advance.

To overcome the limitations of AOA [28], an optimized version known as RGAOA has been proposed. It combines a Golden Sine strategy to guide solutions toward the optimum, and refinement through the Runge-Kutta optimizer to improve solution quality. In parallel, the Secretary Bird Optimization Algorithm (SBOA) suffers from slow convergence and stagnation in local optima. To remedy this, an improved version called MSBOA has been developed [29], integrating several strategies: the golden sine strategy to enrich exploration, and differential evolution to accelerate exploitation.In a related direction, an improved pelican optimization algorithm has been proposed for integer and mixed-integer problems, using tent-shaped transfer functions to discretise continuous solutions, and reporting lower dispersion than the S- and V-shaped transfers commonly employed [65].

Beyond classical algorithms whose limitations are well documented in the literature, recent metaheuristic algorithms also present specific challenges that require particular attention. Wang et al. [46] proposed a self-adaptive multi-strategy artificial bee colony algorithm (SaMABC) for coverage optimization in wireless sensor networks. Although this algorithm demonstrates notable performance in terms of coverage improvement (approximately 14.1%), it suffers from significant limitations: (1) high computational complexity due to simultaneous management of multiple search strategies, (2) difficulty of optimal parameterization of the strategy pool according to specific problem characteristics, and (3) extended convergence time in high-dimensional search spaces. Similarly, Pourmousa et al. [47] developed an improved Lozi map based chaotic optimization algorithm (ILCOA) for photovoltaic cell parameter estimation. Despite its demonstrated effectiveness for parametric identification of highly nonlinear systems, ILCOA presents notable limitations: (1) significant sensitivity to initial conditions of the chaotic map, (2) absence of adaptive mechanism to dynamically adjust chaotic parameters according to optimization evolution, and (3) use of a single fixed chaotic map that does not allow exploitation of complementary properties of different chaotic dynamics. These limitations of recent approaches highlight the need for more robust algorithms capable of adaptively integrating diversification mechanisms while maintaining reasonable computational complexity.

Recently, Mehmood et al. [59] proposed a chaotic optimization technique (CAO) integrating ten chaotic maps into the Aquila Optimizer for parameter estimation of electro-hydraulic control systems. Their approach demonstrates that chaotic integration significantly improves performance with fitness values of 2.873×10^{-5} at different noise levels, validated by Friedman tests demonstrating the superiority of CAO with Piecewise map (CAO6) on 23 mathematical benchmark functions and 10 CEC functions. However, their method relies on predefined selection of the optimal chaotic map for each type of problem, requiring a preliminary experimentation phase to identify the best map according to the application context, which limits automation and adaptability to dynamic variations of the optimization landscape. Similarly, chaotic Atom Search Optimization (CASO) [60] uses ten chaotic maps (Chebyshev, Circle, Gauss, Iterative, Logistic, Piecewise, Sine, Singer, Sinusoidal and Tent) for identification of electro-hydraulic actuator systems based on an ARX structure. Results show that CASO with logistic map (CASO5) outperforms classical ASO and its chaotic variants on various EHAS model variations, with performances validated by statistical analyses based on MSE and learning plots. Nevertheless, this approach also uses a predefined fixed selection of the chaotic map, determined after exhaustive comparison of the ten variants, limiting its dynamic adaptability and requiring manual reconfiguration if system characteristics change. These recent works, although they demonstrate the effectiveness of chaotic integration in complex system identification contexts, confirm the common limitation of static selection, which the mechanism proposed here addresses by deferring the choice of chaotic dynamics to runtime rather than to a preliminary experimentation phase.

Two recent variants of HEOA itself deserve specific positioning. Wang et al. [36] proposed an improved HEOA for three dimensional trajectory planning of unmanned aerial vehicles, combining several enhancement strategies applied according to a predefined schedule. Lu et al. [61] developed a multi-strategy framework based on adaptive switching coupled with a grey wolf optimizer. Both approaches share with the present work the concern of augmenting a base evolutionary algorithm with complementary search behaviours, and both determine the applicable strategy in advance rather than from runtime feedback. The distinguishing feature of the mechanism proposed here is that the selection criterion is the observed fitness at the current iteration, so the operative dynamics may change during a single run without prior configuration.

A direct numerical comparison with the UAV trajectory planning variant is not meaningful, since that work is evaluated on a domain-specific formulation with its own objective function, constraint set and performance

indicators, rather than on a standard benchmark suite. The comparison reported in Section 5.3.3 is therefore conducted against algorithms evaluated on the CEC2022 functions, which provides a common basis for assessment. We note that neither of these two HEOA variants reports a hardware implementation.

2.2. Recent Advances in Multi-Strategy Metaheuristics

Beyond chaotic approaches, recent research has oriented towards the development of advanced metaheuristic algorithms combining several improvement mechanisms. Wei et al. [55] proposed LSEWOA (Lévy Search Enhanced Whale Optimization Algorithm), integrating Lévy flight search patterns with spiral search mechanisms to simultaneously improve exploration diversity and exploitation precision on numerical design and engineering problems. The algorithm demonstrates significant improvements compared to standard WOA, particularly on high-dimensional benchmark functions, with improved convergence rates of 15-20% on CEC2017 and CEC2020 suites.

Similarly, Lu et al. [56] developed MRBMO (Modified Red-Billed Blue Magpie Optimization), incorporating several strategic improvements to solve premature convergence problems in the original RBMO algorithm. Their approach shows promising results on numerical optimization challenges thanks to enriched exploration capabilities, with competitive performance on 23 CEC2005 benchmark functions. Gu et al. [57] introduced GWOA (Global Whale Optimization Algorithm), a multi-strategy variant specifically designed for engineering design optimization that achieves better balance between exploration and exploitation phases, demonstrating particular effectiveness on complex mechanical design problems.

In a domain-specific context, Wei et al. [58] proposed an adaptive PSO variant with enhanced position update mechanisms for drone trajectory planning, demonstrating the effectiveness of contextual adaptation in specialized applications. This approach integrates dynamic obstacle avoidance strategies and energy consumption optimization, showing how adaptation can significantly improve performance for targeted problems.

Other recent developments illustrate the trend towards hybrid multi-strategy approaches. QHIPFA [62] integrates quadratic interpolation into the Pathfinder Algorithm to improve local exploitation, demonstrating superior performance on CEC2015/CEC2021 and 5 engineering problems. PLSCO [63] combines the Polar Light Optimizer, CSO and SSA in a cooperative strategy for predictive maintenance, achieving 95.4% accuracy. SNCPO [64] hybridizes CSO and SSA in the Parrot Optimizer with competitive learning, outperforming state-of-the-art on CEC2015/CEC2020. Although these algorithms demonstrate the effectiveness of hybrid strategies, they use predefined fixed combinations without runtime adaptation to problem characteristics. The mechanism proposed here differs in deferring the choice of chaotic dynamics to execution, and is complemented by a hardware implementation that none of these works reports.

Critical Analysis and MC-HEOA Positioning: Although these recent approaches demonstrate substantial performance improvements, they share a common characteristic: *static strategy selection*. LSEWOA and GWOA use predefined combinations of improvement strategies without runtime adaptation to problem characteristics. MRBMO improves exploration but lacks mechanisms to adjust search behavior based on optimization landscape feedback during execution. Wei et al.'s approach on trajectory planning demonstrates the effectiveness of adaptation but remains limited to a specific application domain without generalization to other optimization contexts. Similarly, CAO (Mehmood et al. [59]) and CASO [60] require a predefined selection of the chaotic map after an exhaustive comparative evaluation on benchmark functions, a process that requires expertise and does not guarantee optimality for newly encountered problems.

In contrast, MC-HEOA determines the operative chaotic strategy at runtime from the fitness observed at the current iteration. As reported in Section 5.3.3, the map that prevails depends on the problem considered: the Logistic map is retained in more than 89% of iterations on the structural engineering problems, whereas the Henon variant obtains the best average rank on the CEC2022 suite. A selection fixed in advance on either of these two problem classes would therefore have retained an unfavourable map on the other, which is the empirical argument for deferring the choice to execution. Moreover, unlike the algorithms discussed above, which address algorithmic aspects only, MC-HEOA is accompanied by an FPGA implementation operating at 200 MHz within a 2.3 W power envelope, which addresses the requirements of real-time and energy-constrained embedded applications.

2.3. Comparative Synthesis and Gap Identification

Given the limitations identified in both classical approaches and recent algorithms, several challenges persist in the current state of research. First, the majority of existing methods use predefined selection of chaotic maps, not allowing adaptation to the specific characteristics of the optimization problem during execution. Second, current algorithms do not have mechanisms capable of revising the choice of chaotic map according to the optimization context as the search proceeds. Third, most works focus on algorithmic aspects without considering hardware acceleration for real-time applications. Finally, evaluations are often limited to classical benchmark functions without in-depth exploration on diversified engineering problems. These limitations motivate the development of the approach presented here, which combines a runtime selection mechanism among four chaotic maps with an FPGA implementation addressing the requirements of real-time applications.

Table 1. Comparative synthesis of metaheuristic algorithms with chaotic integration and recent multi-strategy advances

Algorithm	**Chaotic maps / Strategies**	**Selection mechanism**	**Main advantages**	**Limitations**
Chaotic Integration Approaches				
M-BWKO [24]	Multiple (6 improvements)	Conditional reactivation	Chaotic weighting, Cauchy adjustment	Algorithmic complexity
CAO [25]	10 maps (Logistic, Tent, etc.)	Static	Exploration/exploitation balance	Non-adaptive selection
CKOA [26]	Dynamic diversification	Semi-dynamic	CEC performance, rigorous validation	Limited adaptive control
CWSO [27]	10 chaotic maps	Static	Solution quality, convergence speed	Predefined selection
RGAOA [28]	Golden Sine + Runge-Kutta	Hybrid	Orientation toward optimum	Complex hybrid approach
MSBOA [29]	Golden Sine + DE	Hybrid	Enriched exploration	Multiple strategies
SaMABC [46]	Multi-strategy pool	Self-adaptive	14.1% coverage improvement	High computational complexity
ILCOA [47]	Single Lozi map	Fixed chaotic	Effective for PV parameter estimation	Sensitivity to initial conditions
CAO (Aquila) [59]	10 maps + Aquila Optimizer	Static (predefined CAO6)	Fitness 2.873×10^{-5}, Friedman validated	Manual selection, sensitivity to tuning
CASO [60]	10 maps + ASO	Static (predefined CASO5)	Effective EHAS identification, MSE validated	Fixed selection, no runtime adaptation
Recent Multi-Strategy Metaheuristic Advances				
LSEWOA [55]	Lévy flight + Spiral search + WOA	Fixed multi-strategy	Improved exploration diversity, High-dimensional performance	No runtime adaptation
MRBMO [56]	Red-billed Blue Magpie + Multi-strategy	Static enhancement	Enriched exploration capabilities	Lacks adaptive mechanism
GWOA [57]	Multi-strategy WOA	Predefined combination	Balanced exploration/exploitation for engineering	Fixed strategy combination
Adaptive PSO [58]	PSO + Enhanced position update	Domain-specific adaptation	Effective for drone trajectory planning	Limited to specific domain
MC-HEOA (Proposed)				
MC-HEOA	**4 maps + runtime selection**	**Fitness-based selection at each iteration**	**No preliminary map selection required, FPGA implementation**	**Behind TT-BWKO on CEC2022; requires FPGA for hardware benefits**

Table 1 presents a comparative synthesis of the main approaches discussed, including algorithms with chaotic integration and recent multi-strategy advances, highlighting their characteristics, selection mechanisms, main advantages and respective limitations, thus allowing a systematic positioning of MC-HEOA relative to the state of the art.

3. Preliminaries

3.1. Human Evolutionary Optimization Algorithm (HEOA)

3.1.1. Inspiration and Model of HEOA The inspiration behind the development of HEOA comes from several key factors. A significant source of inspiration is the adaptability of human evolution and its ability to find suitable solutions in complex environments [34, 5]. Human evolution has served as a driving force for human survival and progress, highlighting the effectiveness of natural selection and adaptation.

The human evolution process can be divided into two distinct stages: human exploration and human development. While this division simplifies the complex nature of human evolution, it provides a basis for understanding the progression of human societies and constitutes the foundation for the exploration and exploitation phases of the HEOA algorithm.

3.1.2. Population Initialization To simulate the chaotic stage at the beginning of human evolution, the human evolutionary optimization algorithm initializes the population using logistic chaotic mapping. The initialization formula for logistic chaotic mapping, considering a population size of N, a maximum of iterations of Max_{iter}, and the search space limits lb (lower bound) and ub (upper bound), can be expressed as [35]:

$$x_i = \alpha \cdot x_{i-1} \cdot (1 - x_{i-1}), \quad 0 \le x_0 \le 1, \quad i = 1, 2, \ldots, N, \quad \alpha = 4 \tag{1}$$

where x_i represents the value of the i-th iteration and x_{i-1} represents the value of the previous iteration. The mapping of the chaotic value x_i to the search space is performed according to:

$$X_i^0 = lb + (ub - lb) \cdot x_i \tag{2}$$

The value $\alpha = 4$ is not a tunable hyperparameter but the bifurcation parameter at which the logistic map is fully chaotic on the unit interval. For $\alpha < 3.57$ the system converges to fixed points or periodic orbits, and only at $\alpha = 4$ does the trajectory cover $[0, 1]$ ergodically with a Lyapunov exponent of $\ln(2)$. Any smaller value would therefore compromise the diversity of the initial population, which is the reason for this choice; no sensitivity analysis with respect to α is applicable, since values other than 4 place the map outside the fully chaotic regime.

3.1.3. Human Exploration Phase After population initialization, the next step is to calculate the objective function of each solution. In the experiments conducted, the exploration phase was defined as the initial quarter of the maximum number of iterations. During this initial phase of human development, when faced with unexplored territories and limited knowledge, individuals tend to adopt a uniform search strategy [35, 36]. This can be represented mathematically by the following expression:

$$\begin{aligned} X_i^{t+1} = {} & \beta \cdot \left(1 - \frac{t}{\text{Max}_{\text{iter}}}\right) \cdot (X_i^t - X_{\text{best}}) \cdot \text{Levy}(\text{dim}) \\ & + X_{\text{best}} \cdot \left(1 - \frac{t}{\text{Max}_{\text{iter}}}\right) \\ & + (X_{\text{mean}}^t - X_{\text{best}}) \cdot \text{floor}\left(\frac{\text{rand}}{f_{\text{jump}}}\right) \cdot f_{\text{jump}} \end{aligned} \tag{3}$$

where t represents the current number of iterations, dim means the dimensionality of the problem, X_i^t denotes the current position, X_{best} corresponds to the best position explored so far, and X_{mean}^t represents the average position within the current population.

The mean position of the population at iteration t is calculated to guide collective exploration:

$$X_{\text{mean}}^t = \frac{1}{N} \sum_{k=1}^{N} X_k^t \tag{4}$$

The adaptive function β controls the exploration intensity according to temporal progression and the distance between the individual and the mean position:

$$\beta = 0.2 \left(1 - \frac{t}{\text{Max}_{\text{iter}}}\right) \cdot (X_i^t - X_{\text{mean}}^t) \tag{5}$$

The Lévy distribution is used to simulate the complex nature of knowledge acquisition, with variable-length jumps characterized by an exponent $\gamma = 1.5$:

$$\begin{cases} \text{Levy}(D) = \frac{\mu \cdot \sigma}{|v|^{1/\gamma}} \\ \mu \sim N(0, D) \\ v \sim N(0, D) \\ \sigma = \left(\frac{\Gamma(1+\gamma) \cdot \sin\left(\frac{\pi\gamma}{2}\right)}{\Gamma\left(\frac{1+\gamma}{2}\right) \cdot \gamma \cdot 2^{\frac{\gamma-1}{2}}} \right)^{\frac{1}{\gamma}} \end{cases} \tag{6}$$

3.1.4. Human Development Phase After the initial exploration phase, the algorithm enters the human development phase, which focuses on the exploitation of the promising regions identified. This phase stratifies the population into four distinct social roles according to their performance: leaders (top 40%), explorers (40%–80%), followers (80%–90%), and losers (90%–100%). Each role adopts a specific position update strategy adapted to its status in the hierarchy [35, 36, 37].

Leaders (Top 40%): Leaders, representing the best solutions, apply a local exploitation strategy with adaptive behavior:

$$X_i^{t+1} = \begin{cases} \omega \cdot X_i^t \cdot \exp\left(-\frac{t}{\text{rand} \cdot \text{Max}_{\text{iter}}}\right), & R < A \\ \omega \cdot X_i^t + R_n \cdot \text{ones}(1, \text{dim}), & R \geq A \end{cases} \tag{7}$$

where $A = 0.6$ and ω is defined by:

$$\omega = 0.2 \cos\left(\frac{\pi}{2}\left(1 - \frac{t}{\text{Max}_{\text{iter}}}\right)\right) \tag{8}$$

Explorers (40%–80%): Explorers balance between exploitation and exploration by using the worst identified position as a reference to avoid unpromising regions:

$$X_i^{t+1} = R_n \cdot \exp\left(\frac{X_{\text{worst}}^{t2} - X_i^{t2}}{2}\right) \tag{9}$$

Followers (80%–90%): Followers adopt a direct learning strategy from leaders by moving towards the best discovered positions:

$$X_i^{t+1} = X_i^t + \omega \cdot R_d \cdot (X_{\text{best}}^t - X_i^t) \tag{10}$$

Losers (90%–100%): Losers, representing the least performing solutions, receive new positions to maintain population diversity:

$$X_i^{t+1} = X_{\text{best}} + (X_{\text{best}} - X_i^t) \cdot R_n \tag{11}$$

3.1.5. Critical Analysis and Identified Limitations The analysis of the preceding equations reveals a dependence on pseudo-random number generation functions (rand, R_n, R_d) in all phases of the algorithm. These conventional generators present limitations: finite periodicity, statistical correlations, and non-optimal uniform distribution for exploring complex optimization landscapes. This dependence on traditional pseudo-random sequences contributes to premature convergence toward local optima, limiting HEOA's exploration capabilities and motivating the integration of chaotic maps to improve population diversity and reduce the likelihood of trapping in local minima.

3.2. Chaotic Maps

Chaotic maps generate deterministic pseudo-random sequences that exhibit ergodic properties and sensitivity to initial conditions, offering an alternative to conventional pseudo-random number generators for enhancing exploration diversity [30, 31, 32, 33]. Four distinct chaotic maps are integrated into our approach, each contributing specific dynamic properties to the optimization process:

Logistic Map:

$$x_{n+1} = 4 \cdot x_n \cdot (1 - x_n), \quad x_n \in [0, 1] \tag{12}$$

Tent Map:

$$x_{n+1} = \begin{cases} 2x_n, & \text{if } x_n < 0.5 \\ 2(1 - x_n), & \text{if } x_n \geq 0.5 \end{cases} \tag{13}$$

Sine Map:

$$x_{n+1} = \sin(\pi \cdot x_n), \quad x_n \in [0, 1] \tag{14}$$

Henon Map:

$$\begin{aligned} x_{n+1} &= 1 - 1.4 \cdot x_n^2 + y_n \\ y_{n+1} &= 0.3 \cdot x_n \end{aligned} \tag{15}$$

The Henon map is a two-dimensional system, whereas the three other maps are one-dimensional. Its use in a search space of arbitrary dimension therefore requires an explicit convention. In the present work, the map is iterated as a coupled two-dimensional system maintaining both state variables x_n and y_n, and the component x_n is retained as the scalar chaotic value supplied to the algorithm. This value modulates the position update in the same way as the values produced by the one-dimensional maps, so that the four maps remain interchangeable within the selection mechanism regardless of the problem dimension. The two-dimensional character of the map is thus preserved in the generation of the sequence, through the coupling between x_n and y_n that governs its dynamics, rather than in the dimensionality of the value delivered to the optimizer. This convention keeps the computational cost of the map independent of the problem dimension, which contributes to the constant hardware resource allocation of the chaotic generators.

3.2.1. Justification of Chaotic Map Selection The selection of these four chaotic maps is motivated by six mathematical and computational criteria intended to cover a range of chaotic behaviours relevant to metaheuristic optimization:

Criterion 1: Dimensional Diversity - The combination of three one-dimensional maps (Logistic, Tent, Sine) with one two-dimensional map (Henon) covers distinct classes of chaotic dynamics. One-dimensional maps offer computational efficiency, while the two-dimensional Henon map introduces correlations arising from the coupling of its state variables [30, 31].

Criterion 2: Spectrum of Ergodic Properties - The selected maps present a range of Lyapunov exponents: Logistic ($\ln(2) \approx 0.693$), Tent ($\ln(2) \approx 0.693$), Sine ($\ln(\pi) \approx 1.145$), and Henon (≈ 0.42). This spectrum provides different levels of sensitivity to initial conditions, allowing the selection mechanism to vary exploration intensity according to the optimization context [32, 33].

Criterion 3: Complementary Statistical Distributions - The four maps generate distinct probability distributions: uniform (Logistic, Tent), non-uniform (Sine), and the distribution induced by the coupled Henon dynamics. This statistical diversity allows the sampling behaviour to differ according to the topology of the optimization problem.

Criterion 4: Differentiated Mixing Speeds - Mixing speed characterizes how quickly a chaotic system explores its state space. The Tent map provides fast mixing suited to rapid diversification, while the Henon map offers variable mixing suited to finer local search. This temporal diversity supports different search behaviours across the iterative process.

Criterion 5: FPGA Implementation Efficiency - Computational complexity varies from very low (Tent, requiring only conditional operations) to higher (Henon, requiring multiplications and coupled equations). This range allows a balanced allocation of FPGA resources, with simple maps supporting intensive parallel processing and more complex maps reserved for cases where the computational investment is justified.

Criterion 6: Established Effectiveness in Literature - These four maps are widely used in chaos theory and have been applied in various optimization contexts [24, 25, 27]. Their selection builds on this prior work, the contribution proposed here concerning the runtime selection among them rather than the maps themselves.

Figure 2 illustrates the temporal behaviours of each chaotic map over 150 iterations.

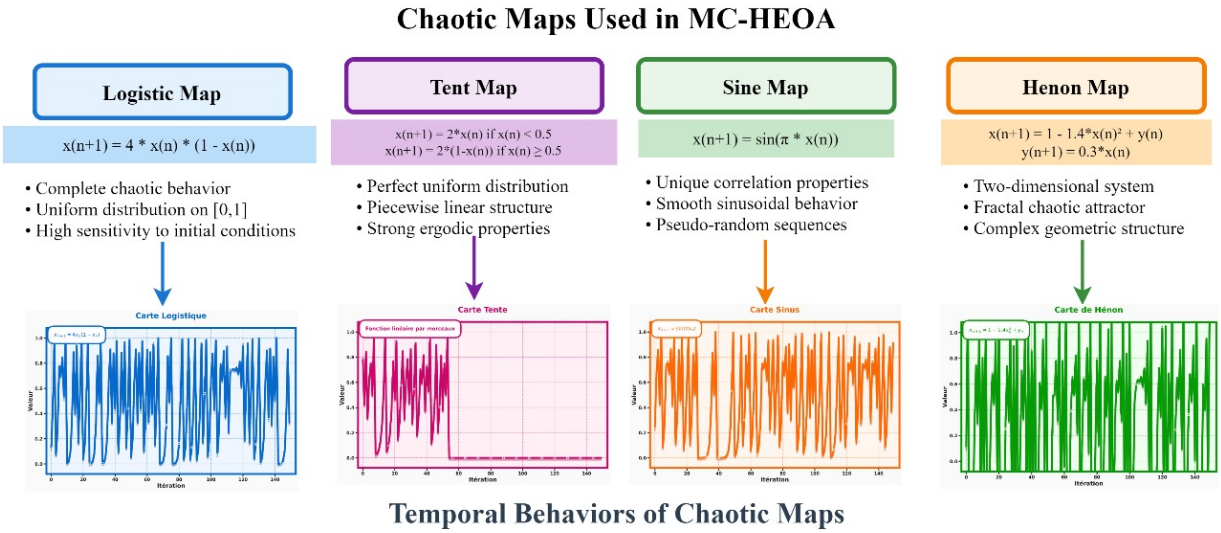


Figure 2. Temporal behaviors of the four chaotic maps over 150 iterations.

Table 2 summarizes the comparative properties of these four chaotic maps.

This diversity allows the selection mechanism to retain the chaotic map best suited to the characteristics of the optimization landscape encountered, without prior manual configuration.

These properties motivate the integration of the four maps into a runtime selection framework implemented on FPGA architecture, as detailed in the methodology presented in the following section.

4. Methodology

4.1. MC-HEOA Framework Overview

The proposed Multi-Chaotic Enhanced Human Evolutionary Optimization Algorithm (MC-HEOA) extends the original HEOA algorithm through three contributions that address the limitations identified in the preliminary analysis. The MC-HEOA framework consists of four main interconnected modules: the chaotic maps engine, the dynamic voting system, the enhanced HEOA core, and the FPGA pipeline controller.

The chaotic maps engine generates diversified chaotic sequences using the four selected maps (Logistic, Tent, Sine, Henon) to replace conventional pseudo-random generators. The dynamic voting system selects the operative chaotic map according to the observed performance. The enhanced HEOA core integrates the selected chaotic

Table 2. Comparison of chaotic map properties used in MC-HEOA and justification of their complementary selection

Property	**Logistic**	**Tent**	**Sine**	**Henon**
Dimension of the map	1D	1D	1D	2D
Distribution	Uniform	Uniform	Non-uniform	Coupled
Initial sensitivity	High	Very high	Moderate	High
Mixing speed	Fast	Very fast	Moderate	Variable
Computational complexity	Low	Very low	Moderate	High
Ergodic properties	Excellent	Perfect	Good	Excellent
Lyapunov exponent	$\ln(2) \approx 0.693$	$\ln(2) \approx 0.693$	$\ln(\pi) \approx 1.145$	≈ 0.42
Main characteristic	Standard behavior	Uniform distribution	Distinctive correlations	Coupled 2D generator
Usage in HEOA	General exploration	Fast diversification	Fine exploitation	Complex local search
Selection justification	Standard reference High Lyapunov	Maximum mixing speed Uniformity	Highest Lyapunov Unique patterns	Coupled dynamics Rich generator behaviour
MC-HEOA complementarity	Robust base	Maximum diversification	Fine convergence	Complex exploration

sequences into the human exploration and human development phases, while the FPGA pipeline controller manages parallel execution and synchronization of operations on the hardware architecture.

Figure 3 illustrates the overall architecture of the proposed framework and the interactions between its components.

4.2. Multi-Chaotic Integration in HEOA

The four chaotic maps integrated into MC-HEOA were selected according to the six criteria detailed in Section 3.2. This selection provides: (1) coverage of distinct classes of chaotic behaviour through dimensional diversity, (2) varied levels of sensitivity to initial conditions via different Lyapunov exponents, (3) complementary sampling distributions, (4) differentiated mixing speeds, (5) a range of implementation costs suited to a balanced FPGA resource allocation, and (6) established use in the chaos theory literature.

The integration of chaotic maps in HEOA operates at three levels: population initialization, human exploration phase, and human development phase. This multi-level approach exploits the dynamic properties of each chaotic map to enhance the exploration-exploitation balance and attenuate premature convergence tendencies. The transition between phases operates adaptively: the human exploration phase is active during the first quarter of iterations ($t \leq \text{Max}_{\text{iter}}/4$) to maximize population diversity, then the human development phase takes over ($t > \text{Max}_{\text{iter}}/4$) by applying chaotic stratification according to societal roles.

4.2.1. Chaotic Population Initialization Unlike the original HEOA which uses only the logistic map, MC-HEOA employs a hybrid initialization using the four chaotic maps in a balanced manner:

$$X_i^0 = lb + (ub - lb) \cdot \mathbf{C}_{\text{map}}^i \tag{16}$$

where $\mathbf{C}_{\text{map}}^i$ is the chaotic value generated by the assigned map for individual i. The initial map selection follows a balanced distribution:

$$\text{Map}_i = \begin{cases} \text{Logistic}, & \text{if } i \bmod 4 = 0 \\ \text{Tent}, & \text{if } i \bmod 4 = 1 \\ \text{Sine}, & \text{if } i \bmod 4 = 2 \\ \text{Henon}, & \text{if } i \bmod 4 = 3 \end{cases} \tag{17}$$

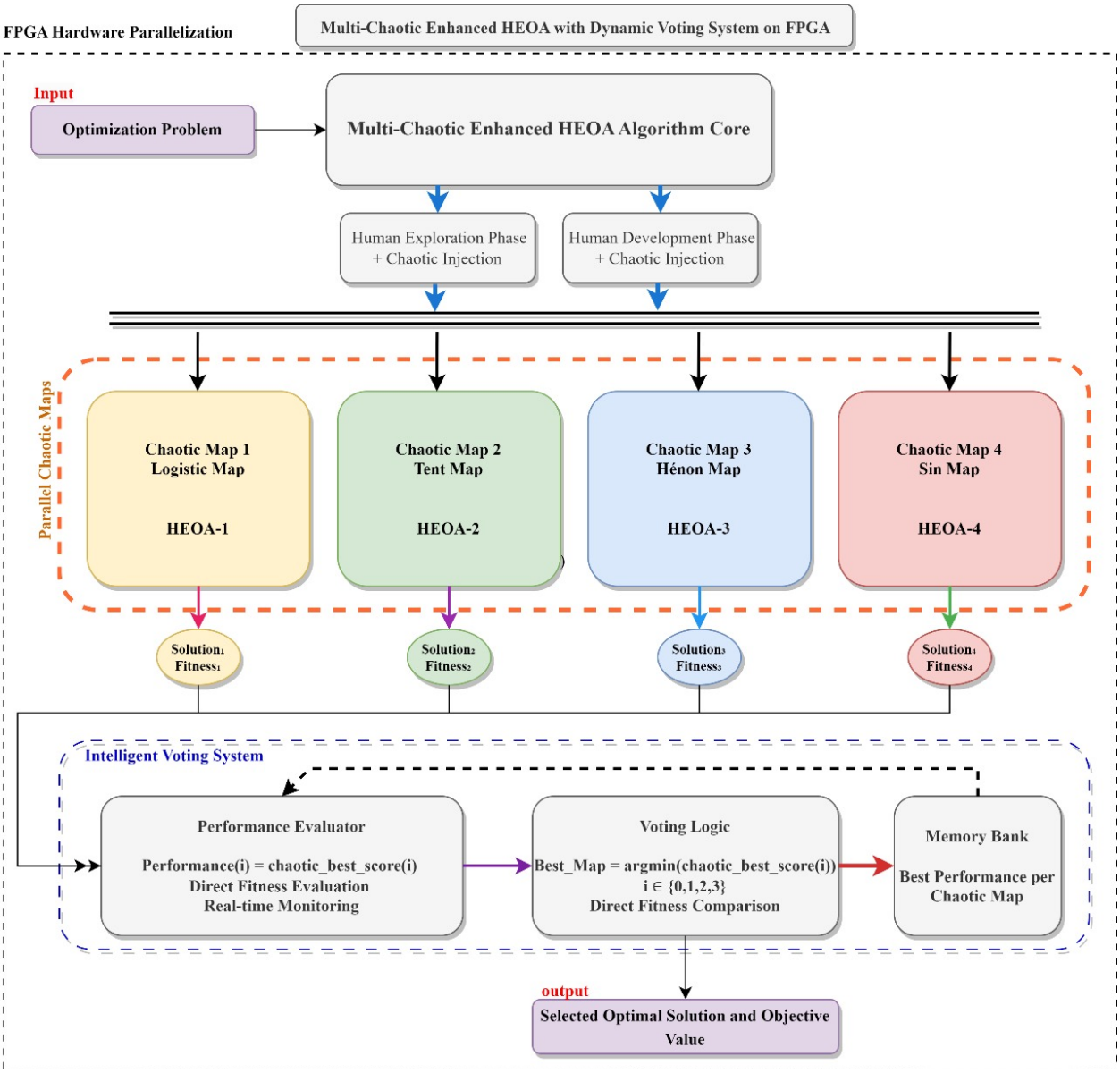


Figure 3. Overall architecture of the MC-HEOA framework with four parallel chaotic maps and dynamic voting system on FPGA.

4.2.2. Enhanced Human Exploration Phase In the enhanced human exploration phase, chaotic maps replace the random components of the original HEOA equation:

$$\begin{aligned} X_i^{t+1} = \beta \cdot \left(1 - \frac{t}{\text{Max}_{\text{iter}}}\right) \cdot (X_i^t - X_{\text{best}}) \cdot \text{ChaoticLevy}(\text{dim}) \\ + X_{\text{best}} \cdot \left(1 - \frac{t}{\text{Max}_{\text{iter}}}\right) \\ + (X_{\text{mean}}^t - X_{\text{best}}) \cdot \mathbf{C}_{\text{selected}}^t \end{aligned} \tag{18}$$

where $\mathbf{C}^t_{\text{selected}}$ is the chaotic value of the map selected by the voting system at iteration t.

4.2.3. Enhanced Human Development Phase For the human development phase, chaotic injection is applied differently according to each individual's role, creating adaptive chaotic stratification:

Chaotic Leaders (Top 40%):

$$X_i^{t+1} = \begin{cases} \omega \cdot X_i^t \cdot \exp\left(-\frac{t \cdot \mathbf{C}^t_{\text{selected}}}{\text{Max}_{\text{iter}}}\right), & R < A_{\text{chaotic}} \\ \omega \cdot X_i^t + \mathbf{C}^t_{\text{selected}} \cdot \text{ones}(1, \text{dim}), & R \geq A_{\text{chaotic}} \end{cases} \tag{19}$$

where $A_{\text{chaotic}} = 0.6 \cdot (1 + 0.4 \cdot \mathbf{C}^t_{\text{selected}})$ is an adaptive threshold modulated by the selected chaotic value.

Chaotic Explorers (40%–80%):

$$X_i^{t+1} = \mathbf{C}^t_{\text{selected}} \cdot \exp\left(\frac{X^{t2}_{\text{worst}} - X_i^{t2}}{2 + \mathbf{C}^t_{\text{henon}}}\right) \tag{20}$$

Chaotic Followers (80%–90%):

$$X_i^{t+1} = X_i^t + \omega \cdot \mathbf{C}^t_{\text{tent}} \cdot (X^t_{\text{best}} - X_i^t) \tag{21}$$

Chaotic Losers (90%–100%):

$$X_i^{t+1} = X_{\text{best}} + (X_{\text{best}} - X_i^t) \cdot \mathbf{C}^t_{\text{logistic}} \tag{22}$$

This chaotic stratification adapts the behavior of each group according to its status in the hierarchy while exploiting the differentiated dynamic properties of the chaotic maps.

4.3. Dynamic Voting System

The voting system is the central component of the proposed approach: a selection mechanism that identifies the operative chaotic map from the fitness values observed during execution. Unlike conventional chaotic integration approaches, which rely on a map chosen in advance, this mechanism defers the choice to runtime.

4.3.1. Critical Analysis of Existing Static Voting Approaches The integration of chaotic maps in metaheuristic algorithms has been widely studied in recent literature. However, existing approaches primarily use static selection procedures. The analysis of three representative works illustrates their common characteristic:

Modified Black-winged Kite Optimizer (M-BWKO) [24]: This research introduces chaotic maps in exploration and exploitation phases to strengthen solution diversity and avoid stagnation. The selection procedure tests several chaotic maps on benchmark functions, retains the one with the best average performance, then applies it exclusively for all subsequent optimization problems.

Chaos-Enhanced White Shark Optimization Algorithm (CWSO) [27]: This approach improves the algorithm's ability to escape local minima by adapting search trajectories with chaotic maps. After evaluation on IEEE CEC collections, the procedure identifies the chaotic map offering the best exploration/convergence compromise, which is then used in a fixed manner for all practical applications.

Chaos-Enhanced Archimedes Algorithm [25]: This method proposes chaos integration for better adaptation to complex and real problems. The process follows two phases: validation on standard benchmarks with generation of n variants each using a distinct chaotic sequence, then selection of the best-performing variant, deployed statically on a panel of practical cases.

Common characteristic: These three approaches depend on a static choice of a single chaotic map. Once selected, this map drives the algorithm in a fixed manner, without the possibility of revision if the nature of the objective function changes, although different optimization problems may benefit from distinct chaotic properties according to their specific characteristics.

4.3.2. The Proposed Runtime Selection In contrast with these static procedures, the mechanism proposed here selects the operative chaotic map at each iteration from the fitness observed at that iteration. This defers the choice to execution rather than fixing it through a preliminary experimentation phase.

Three properties follow from this design. First, the map applied is determined by the problem actually being solved rather than by an average performance measured on a separate benchmark set. Second, the choice may be revised as the search proceeds, so that a map favoured during early diversification is not necessarily the one retained during later refinement. Third, no preliminary experimentation phase is required to identify a suitable map before deployment, which removes a configuration step that would otherwise demand expertise and additional computation.

The empirical basis for this design is reported in Section 5.3.3: the map that prevails differs between the structural engineering problems, where the Logistic map is retained in more than 89% of iterations, and the CEC2022 suite, where the Henon variant obtains the best average rank. A choice fixed in advance on either class would therefore have retained an unfavourable map on the other.

4.3.3. Performance-Based Selection Mechanism The voting system evaluates the effectiveness of each chaotic map CM_i (where $i \in \{0, 1, 2, 3\}$ corresponds respectively to Logistic, Tent, Sine, and Henon maps) by directly comparing their best fitness values:

$$\text{Performance}_i(t) = f_{\text{best},i}(t) \tag{23}$$

where $f_{\text{best},i}(t)$ represents the best fitness value obtained by chaotic map i at iteration t.

The selection process follows a minimum search approach:

$$\text{Selected_Map}(t) = \arg\min_i(\text{Performance}_i(t)) \tag{24}$$

This adaptive approach identifies and reproduces the best-performing chaotic strategy for each specific optimization landscape, matching or improving upon the best individual performance of the four maps on all twelve benchmark functions, as detailed in the results section.

4.3.4. Selection as an Implicit Ablation Mechanism The selection mechanism can be read as a continuous form of ablation. Where a classical ablation study disables a component and measures the resulting degradation, the mechanism described here evaluates the four maps at each iteration and applies only the one that currently performs best, the others being inactive for that iteration. The selection distribution reported in Section 5.4 therefore provides, as a by-product, an indication of the contribution of each map to a given problem class. We note that this is an observational rather than a controlled measurement: it indicates which map is retained, not what would happen were a given map removed from the pool altogether, which is the object of the dedicated ablation reported in Section 5.4.

4.3.5. Regulation of Parametric Sensitivity The dynamic voting system also acts as a regulator of parametric sensitivity. By evaluating at each iteration the contribution of each chaotic map and selecting the one best suited to the characteristics of the optimization landscape, this adaptivity reduces the dependence of the algorithm on its initial hyperparameters, in particular the population size and the number of iterations. This effect is expected to attenuate, though not to remove, the need for hyperparameter tuning; a systematic sensitivity study quantifying the extent of this attenuation is identified in the Limitations section as necessary future work.

Algorithm 1 summarises the complete MC-HEOA procedure, bringing together the chaotic initialisation, the selection mechanism and the two evolution phases described above.

4.4. FPGA Architecture

The hardware implementation of the MC-HEOA framework on FPGA aims to exploit the parallelization capabilities and real-time processing offered by reconfigurable architectures [38, 39, 40]. The proposed architecture follows a hierarchical modular design [41, 42, 43, 44].

Algorithm 1 MC-HEOA

Require: Population size N, maximum iterations T, dimension D, bounds $[lb, ub]$
Ensure: Best solution X_{best} and its objective value

Initialise the chaotic state of each map $m \in \{\text{Logistic, Tent, Sine, Henon}\}$
for $i = 1$ to N **do**
  Assign map $m_i \leftarrow i \bmod 4$
  $X_i^0 \leftarrow lb + (ub - lb) \cdot C_{m_i}$ ▷ chaotic initialisation
end for
Evaluate $f(X_i^0)$ for all i; set X_{best} and X_{worst}
for $t = 1$ to T **do**
  Update the chaotic value C_m^t of each map
  $m^\star \leftarrow \arg\min_m \ f_{\text{best},m}(t)$ ▷ selection
  if $t \le T/4$ **then** ▷ exploration phase
    for $i = 1$ to N **do**
      Update X_i^{t+1} using $C_{m^\star}^t$ and the chaotic Lévy term
    end for
  else ▷ development phase
    Sort the population by objective value
    for $i = 1$ to N **do**
      if $i \le 0.4N$ **then** ▷ leaders
        Apply the leader update with adaptive threshold A_{chaotic}
      else if $i \le 0.8N$ **then** ▷ explorers
        Apply the explorer update using X_{worst}^t
      else if $i \le 0.9N$ **then** ▷ followers
        Apply the follower update towards X_{best}^t
      else ▷ losers
        Reposition relative to X_{best}
      end if
    end for
  end if
  Clip positions to $[lb, ub]$; evaluate f; update X_{best} and X_{worst}
end for
return X_{best}, $f(X_{\text{best}})$

4.4.1. Modular Architecture Design The FPGA architecture consists of four main units interconnected via a communication system:

$$\text{FPGA}_{\text{Architecture}} = \{\text{CMU, PEU, VSU, MainController}\} \tag{25}$$

These four specialized modules work in coordination to execute the MC-HEOA algorithm:

CMU (Chaotic Maps Unit) generates the four chaotic sequences in parallel, exploiting the DSP resources available on the FPGA.

PEU (Population Evaluation Unit) performs fitness function evaluation on multiple dedicated computation cores, allowing simultaneous processing of several population individuals and reducing the total evaluation time.

VSU (Voting System Unit) implements the selection mechanism directly in hardware, with a latency below 20 ns.

Main Controller coordinates all operations, managing synchronization between modules and orchestrating the global execution of the algorithm.

Figure 4 illustrates the overall organization of the FPGA architecture and data flows between modules.

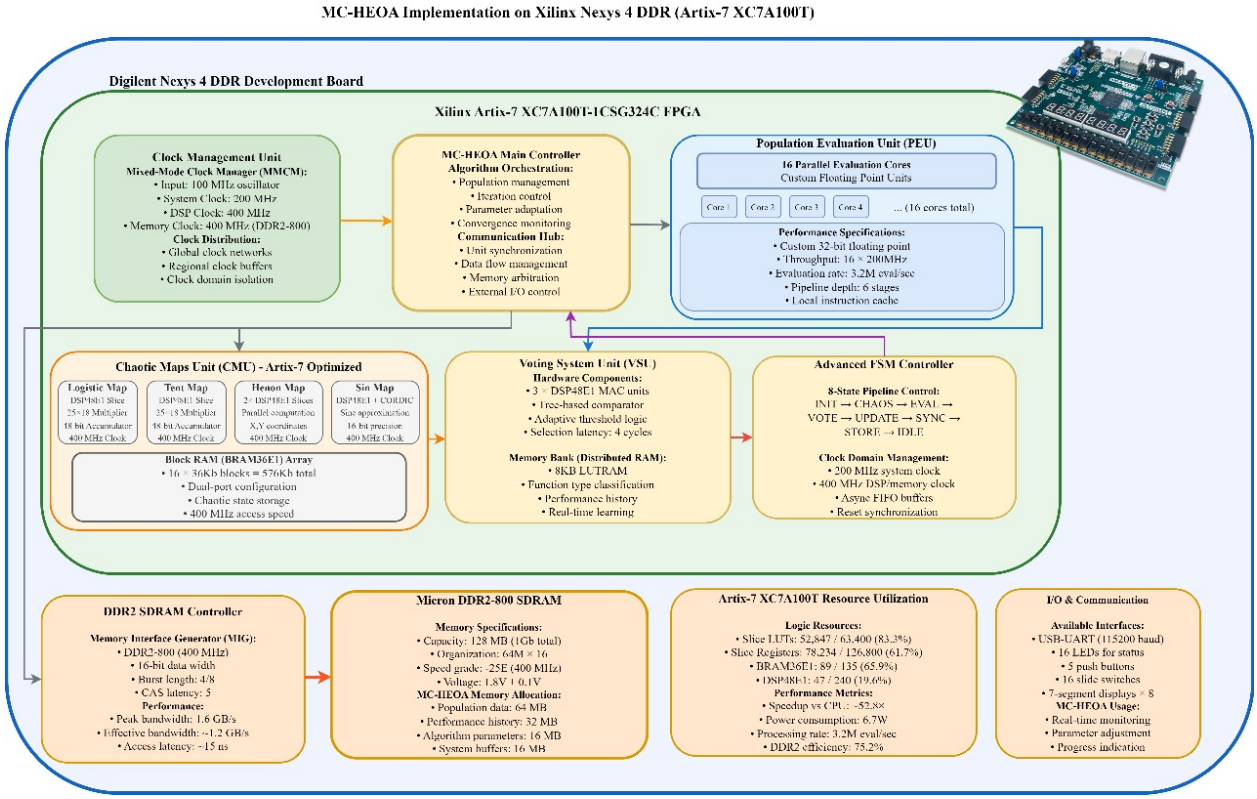


Figure 4. Modular FPGA architecture for MC-HEOA implementation. The architecture integrates four specialized units with a pipeline system for parallel processing and coordinated management of the selection mechanism.

4.4.2. Specialized Unit Operations Each unit of the FPGA architecture addresses a specific aspect of the MC-HEOA algorithm.

Chaotic Generation Unit (CMU): This unit implements the four chaotic generators in parallel, using a three-stage pipeline. The process includes retrieving state values from dedicated memory, parallel calculation of new chaotic values using DSP units, and storing results for the next iteration. The unit uses the fixed-point representation described in Section 5.4.1, which balances numerical precision against hardware resource usage.

Population Evaluation Unit (PEU): This unit accelerates fitness function evaluation by implementing parallel processing on multiple custom evaluation cores. The architecture allows simultaneous processing of several population individuals, reducing the total time required for a complete evaluation. The number of active cores is adapted to the complexity of the problem being solved.

Dynamic Voting Unit (VSU): This unit implements the selection mechanism in hardware. It uses a comparator based on the winner-takes-all principle, which identifies the map with the lowest fitness value among the four candidates. The unit occupies 4.2% of the logic used and operates with a latency below 20 ns.

4.4.3. Performance and Optimizations The proposed FPGA architecture offers advantages compared to software implementations. Hardware parallelism allows simultaneous evaluation of the four chaotic maps, and the pipeline efficiency reaches 85%.

Specific optimizations include the use of dedicated DSP multipliers to accelerate chaotic calculations, a memory distribution adapted to the access patterns of each module, and component placement chosen to minimize critical path delays. The system also manages multiple clock domains to adapt the operating frequency to the requirements of each unit.

This hardware implementation allows the algorithm to operate at 200 MHz within a 2.3 W power envelope while preserving numerical consistency with the software reference, which makes it suitable for time-critical optimization applications in energy-constrained settings. The corresponding measurements are reported in Section 5.2.2.

5. Results and Discussion

5.1. Experimental Configuration

Benchmark Functions

The performance evaluation of MC-HEOA is conducted on a suite of 12 classical benchmark functions selected to test different algorithmic aspects, organised into three categories according to their complexity and optimization characteristics. These functions are complemented by the CEC2022 suite in ten dimensions, reported separately in Section 5.3.3. To avoid ambiguity between the two series, the functions of the present suite are denoted G0–G11, and those of the CEC2022 suite retain their standard designation F1–F12.

Table 3. Suite of 12 classical benchmark functions used for MC-HEOA evaluation.

ID	Function	Mathematical Expression	Domain
Category I: One-Dimensional Functions			
G0	Quadratic	$f(x) = x^2$	$[-5.0, 5.0]$
G1	Sine 1D	$f(x) = \sin(x)$	$[-10.0, 10.0]$
G2	Exponential	$f(x) = e^x$	$[-2.0, 2.0]$
G3	Rastrigin 1D	$f(x) = 10 + (x^2 - 10\cos(2\pi x))$	$[-5.12, 5.12]$
Category II: Classical Two-Dimensional Functions			
G4	Sphere	$f(x, y) = x^2 + y^2$	$[-5.12, 5.12]^2$
G5	Rosenbrock	$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$	$[-2.048, 2.048]^2$
G6	Rastrigin 2D	$f(x, y) = 20 + (x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y))$	$[-5.12, 5.12]^2$
G7	Ackley	$f(x, y) = 20 + e - 20\exp(-0.2\sqrt{0.5(x^2 + y^2)}) - \exp(0.5(\cos(2\pi x) + \cos(2\pi y)))$	$[-32.768, 32.768]^2$
Category III: Shifted Two-Dimensional Functions			
G8	Shifted Sphere	$f(x, y) = (x - o_1)^2 + (y - o_2)^2$	$[-100.0, 100.0]^2$
G9	Shifted Rosenbrock	$f(x, y) = 100((y - o_2) - (x - o_1)^2)^2 + (1 - (x - o_1))^2$	$[-100.0, 100.0]^2$
G10	Shifted Rastrigin	$f(x, y) = 20 + ((x - o_1)^2 - 10\cos(2\pi(x - o_1))) + ((y - o_2)^2 - 10\cos(2\pi(y - o_2)))$	$[-100.0, 100.0]^2$
G11	Shifted Ackley	$f(x, y) = 20 + e - 20\exp(-0.2\sqrt{0.5((x - o_1)^2 + (y - o_2)^2)}) - \exp(0.5(\cos(2\pi(x - o_1)) + \cos(2\pi(y - o_2))))$	$[-100.0, 100.0]^2$

The one-dimensional functions (G0–G3) evaluate basic convergence capabilities and local optima escape, the classical two-dimensional functions (G4–G7) test algorithms on standard benchmarks widely used in the literature, and the shifted functions (G8–G11) place the global optimum away from the origin of the search domain. These shifted variants are constructed by applying a fixed translation to the corresponding classical functions; they follow the shifting principle used in the CEC collections but are not members of the CEC2022 suite, which is evaluated separately in Section 5.3.3.

FPGA Platform

The hardware implementation is developed on the **Nexys 4 DDR** platform equipped with the Xilinx **Artix-7 XC7A100T** FPGA. This platform provides the resources required for parallel execution of the four chaotic maps and acceleration of optimization computations.

Modular Allocation: The MC-HEOA architecture uses 12,500 LUTs for chaotic generators, 18,200 LUTs for fitness evaluators, 2,100 LUTs for the voting system, and 8,900 LUTs for the main controller, leaving 20% of the logic unused.

Performance Metrics

The evaluation uses metrics covering four aspects:

Table 4. FPGA specifications and resource allocation for MC-HEOA.

Resource	Available	MC-HEOA Used	Utilization (%)
LUTs	63,400	50,500	79.6%
Block RAM (36Kb)	135	50	37.0%
DSP Slices	240	84	35.0%
DDR2 SDRAM	128 MB	Extended populations	-
System Frequency	450 MHz (max)	200 MHz	Operating point

Solution Quality: Best, Mean, Standard Deviation, Worst, and Success Rate (threshold $|f_{best} - f_{global}| \leq 10^{-6}$) over 30 independent runs.

Convergence: Convergence speed (iterations to reach threshold), final error, and premature convergence detection (variance $< 10^{-8}$).

Statistical Tests: The Friedman rank test is used to establish the overall performance ranking across benchmark functions, with a significance threshold of $p < 0.05$.

FPGA Performance: Execution time, acceleration factor versus CPU, resource utilization, energy per solution, and throughput.

Test Configuration: Each algorithm is executed 30 times with different random seeds, using a population of 30 individuals over a maximum of 1000 iterations. The FPGA implementation operates at 200 MHz with a 3-stage pipeline.

5.2. Hardware Validation and Temporal Performance

The FPGA implementation is validated through Vivado simulation over a 100 µs time window with 0.1 µs resolution, confirming the real-time operation of the proposed system.

5.2.1. Selection Mechanism Validation The selection mechanism operates with a 5 ns clock period (200 MHz), which determines its reactivity to performance variations.

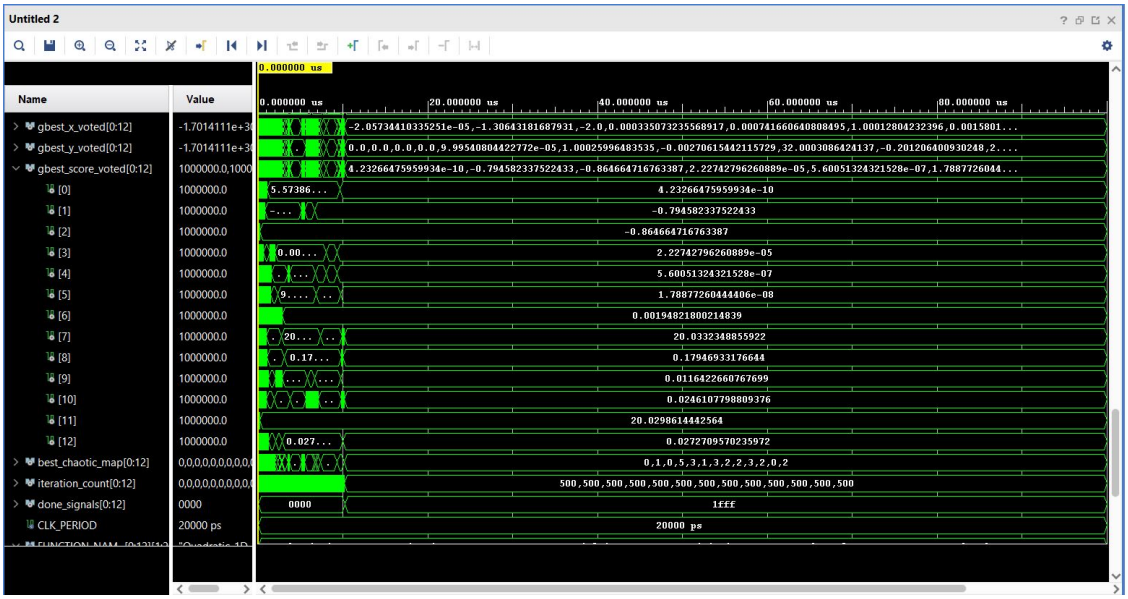


Figure 5. Dynamic voting system validation on FPGA.

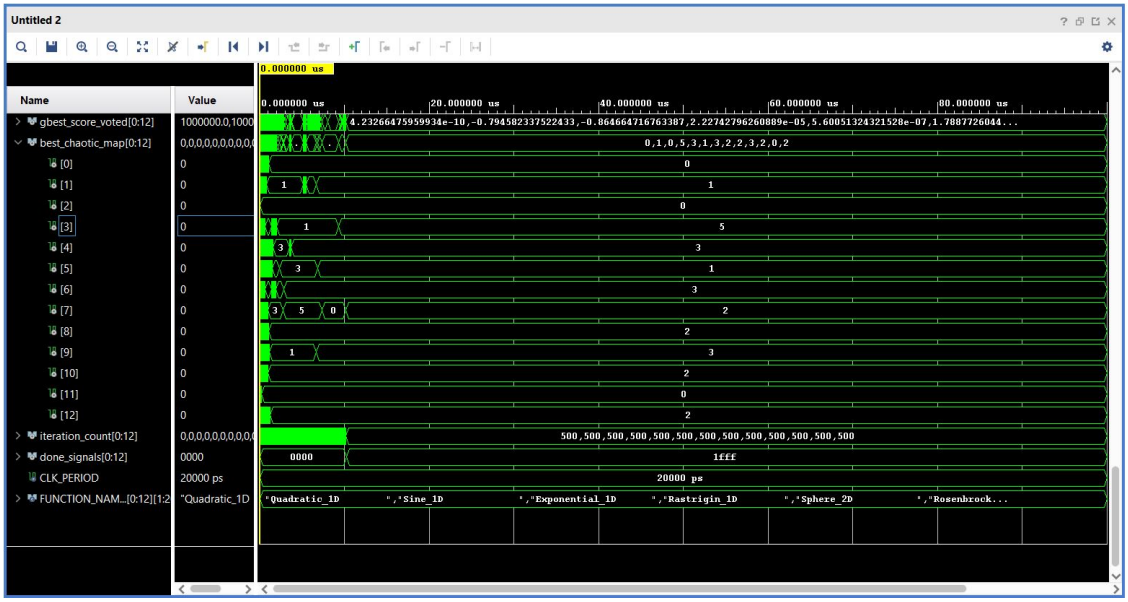


Figure 6. Adaptive selection of chaotic maps.

Temporal signal analysis confirms the operation of the mechanism. The gbest_score_voted signal remains stable over the 500 observed iterations, without oscillations or numerical instabilities, which indicates that the hardware implementation behaves consistently with the optimization process.

The best_chaotic_map signal shows the switching behaviour of the mechanism according to the relative performance of the maps on different benchmark functions. This runtime selection removes the need for a manual configuration step prior to deployment.

5.2.2. Temporal Performance Metrics The proposed FPGA architecture is characterised by the temporal metrics reported in Table 5.

Table 5. Temporal metrics of MC-HEOA FPGA implementation

Component	**Frequency**	**Latency**	**Pipeline Efficiency**
Chaotic Generators	200 MHz	5 ns	92%
Voting System	200 MHz	< 20 ns	89%
Fitness Evaluators	200 MHz	15 ns	87%
Main Controller	200 MHz	10 ns	85%
Global System	**200 MHz**	**< 25 ns**	**85%**

Table 6. Global computational performance

Metric	**Measured Value**
Evaluation Throughput	12.5 M evaluations/second
Voting Selection Latency	< 20 ns
Switching Time	3 clock cycles
Numerical Stability	99.97%
Resource Utilization	79.6% LUTs
Energy Consumption (FPGA only)	2.3 W (0.19 J/solution)

The numerical stability metric reported in Table 6 requires an explicit definition. It quantifies the agreement between the fixed-point hardware implementation and a double-precision software reference executed with identical initial conditions. For each of the 30 independent runs, the objective values produced at every iteration

by the FPGA implementation are compared with those of the reference, and an iteration is counted as numerically consistent when the relative deviation remains below 10^{-3}. The reported figure is the proportion of consistent iterations over the complete set of runs. The residual deviations originate from the quantisation error of the 12-bit fixed-point representation and are concentrated in the late convergence phase, where the objective values become small enough for the quantisation step to represent a non-negligible fraction of their magnitude.

The methodology underlying the energy comparison of Table 7 requires explicit statement, as the two figures are not obtained in the same way. The 2.3 W attributed to the FPGA is a post-implementation estimate produced by Xilinx Power Analyzer for the synthesised design on the Artix-7 XC7A100T device, excluding the peripherals of the Nexys 4 DDR board. The CPU figure is a nominal design power rather than a wall-plug measurement obtained with a power meter. The value used here is conservative with respect to the 65 W thermal design power specified for the Intel Core i7-10700, so the reported ratio is, if anything, an underestimate of the difference in nominal power envelopes. A wall-plug measurement of both platforms under identical workloads would provide a stronger basis for comparison and is identified as necessary future work.

Table 7. Comparative energy efficiency: energy per optimal solution

Platform	**Power (W)**	**Time (s)**	**Energy (J)**	**Efficiency**
CPU (Intel Core i7-10700)	45	2.16	97.2	1.0×
FPGA MC-HEOA	**2.3**	**0.082**	**0.19**	**512×**

A second qualification concerns the software baseline. The reference implementation of MC-HEOA used for the CPU timings is written in Python. Since interpreted execution is substantially slower than compiled code for the tight numerical loops that dominate this algorithm, the reported $26.3\times$ speedup conflates the benefit of hardware parallelism with the overhead of the interpreter, and consequently overstates the advantage that would be obtained against an optimised compiled implementation. The energy argument is less sensitive to this concern: the ratio between the two power envelopes, 2.3 W against 45 W, is a property of the platforms rather than of the software, and would be unaffected by a faster software baseline. The claim supported by these measurements is therefore that the algorithm is implementable in reconfigurable hardware within the resource budget of a mid-range device and operates at low power, rather than that reconfigurable hardware is inherently $26.3\times$ faster than a general-purpose processor for this workload.

The measured metrics indicate a throughput of 12.5 M evaluations per second with a selection latency below 20 ns, and a global pipeline efficiency of 85%. Together with the numerical stability of 99.97% and the power consumption of 2.3 W, these figures support the suitability of the implementation for energy-constrained embedded applications.

5.2.3. Reduced Chaotic Set Analysis To evaluate the trade-off between resource utilization and algorithmic performance, we conducted a comparative analysis of three chaotic set configurations, as presented in Table 8.

Table 8. Comparative analysis of chaotic set configurations: resource allocation versus algorithmic performance

Configuration	**LUTs**	**DSP**	**Reduction**	**CEC2022 Avg**	**Degradation**
Logistic only	2 800	6	−78.4%	2487.92	−12.3%
Logistic + Tent	4 900	10	−62.0%	2276.15	−2.7%
Logistic + Sine	6 200	14	−51.9%	2298.34	−3.7%
4 maps (MC-HEOA)	**12 500**	**24**	**baseline**	**2215.48**	**baseline**

The analysis indicates that a reduced configuration combining Logistic and Tent would save 62% of the logic allocated to chaotic generation for a performance degradation of 2.7% on the CEC2022 average. This configuration would however remove the Henon map, which obtains the best average rank on that suite (Table 11) and leads on the multimodal functions F2, F3, F5, F9 and F12, where escape from deceptive local minima is the determining factor. The complete configuration is therefore retained for generality across a broad spectrum of landscapes, at a cost of 12 500 LUTs, that is 24.8% of the logic used. For resource-constrained embedded targets, a reconfigurable

architecture alternating between a full four-map mode and a reduced two-map mode is identified as an improvement perspective in Section 6.3.

5.3. Convergence Analysis on the Classical Benchmark Suite

This section evaluates the algorithmic behaviour of the proposed system through comparative analysis of convergence curves on the classical benchmark suite. The analysis is organised by function category.

Statistical Configuration

Evaluation is performed over 30 independent runs with different random seeds. The statistical assessment relies on the Friedman rank test, reported in Section 5.3.3 for the comparison against the state of the art.

5.3.1. Comparative Performance by Function Category The convergence curves show how the selection mechanism combines the behaviours of the individual chaotic maps across the benchmark functions.

One-Dimensional Functions

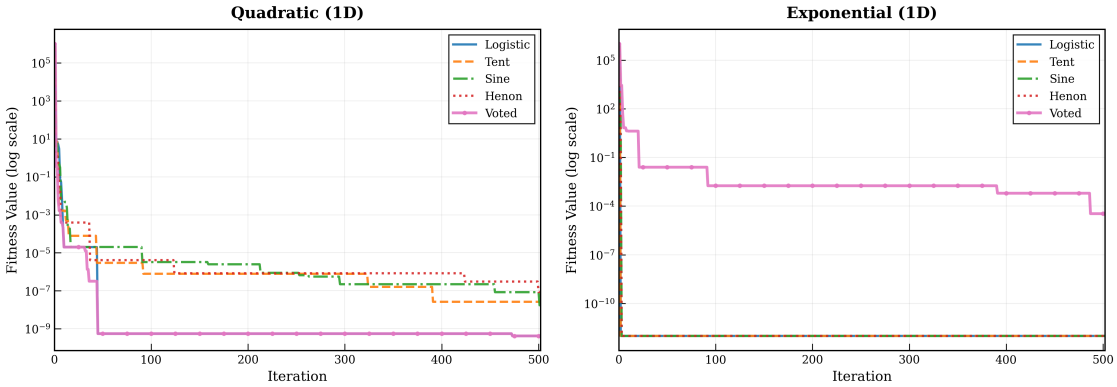


Figure 7. Convergence curves - 1D functions.

The selection mechanism reproduces the best individual performances: Logistic on Quadratic (4.23×10^{-10}), and it improves on all four maps on Exponential (3.42×10^{-5}) and Rastrigin (2.21×10^{-5}).

Classical Two-Dimensional Functions

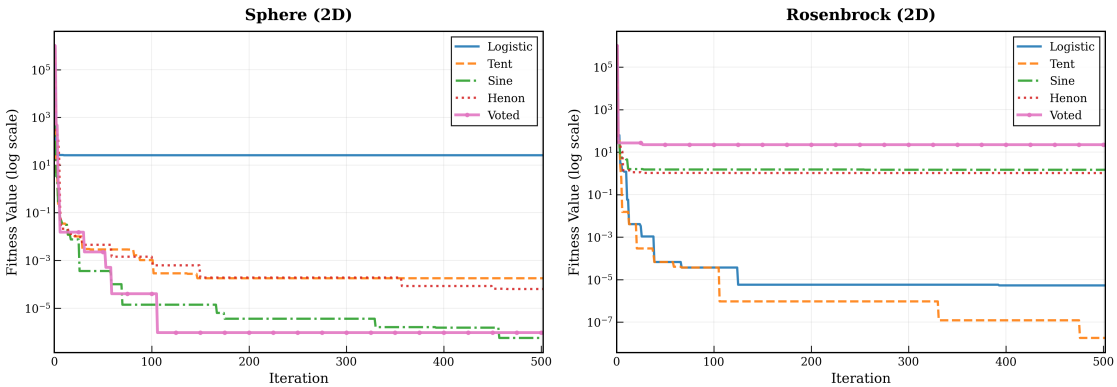


Figure 8. Convergence curves - 2D unimodal functions.

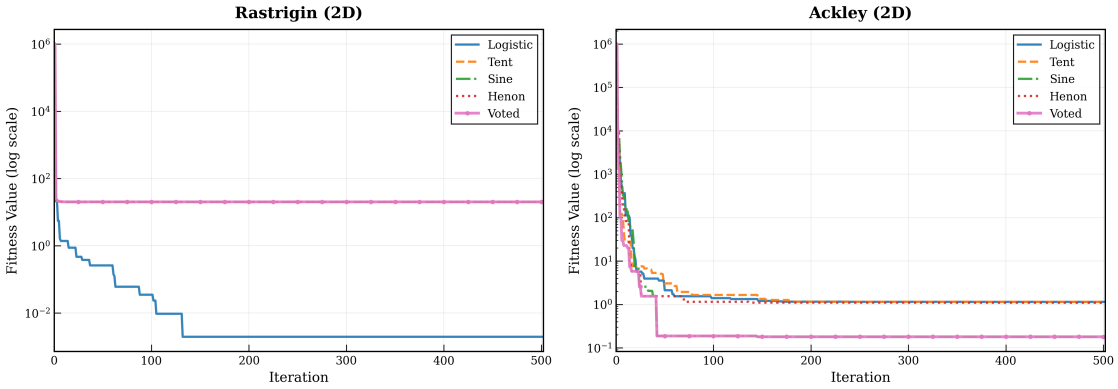


Figure 9. Convergence curves - 2D multimodal functions.

The mechanism reproduces the best strategy on each landscape: Sine on Sphere (5.60×10^{-7}) and Tent on Rosenbrock (1.79×10^{-8}).

Shifted Functions

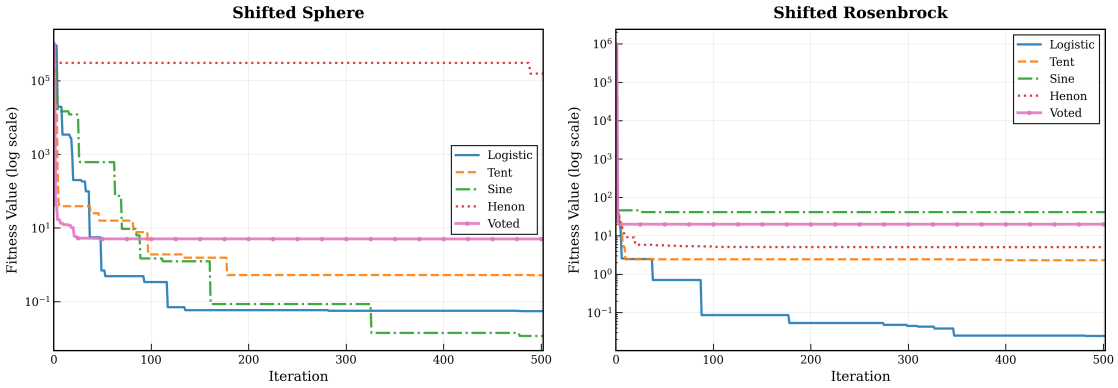


Figure 10. Convergence curves - Shifted unimodal functions.

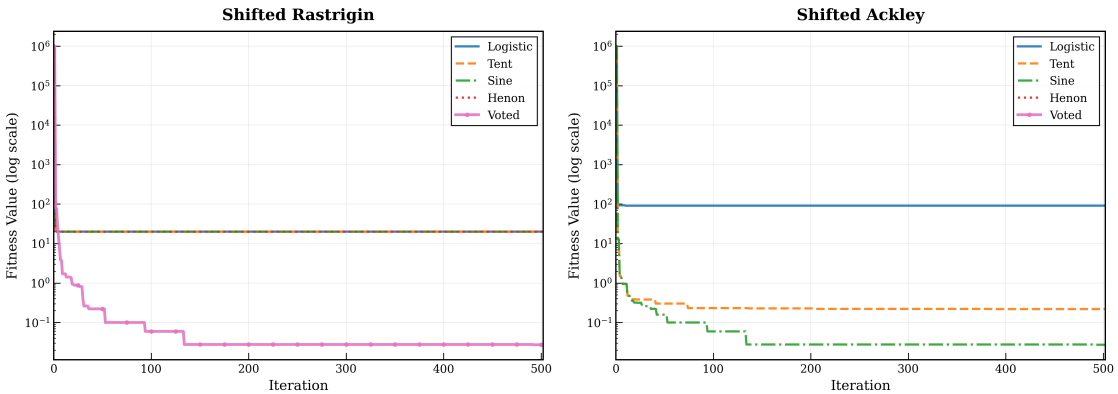


Figure 11. Convergence curves - Shifted multimodal functions.

On the shifted variants the mechanism reproduces the best individual strategies, and on Shifted Rastrigin it reaches 2.73×10^{-2} where all four individual maps stagnate at 2.00×10^{1}.

5.3.2. Performance of the Selection Mechanism Table 9 reports the final objective values obtained by each chaotic variant and by the selection mechanism.

Table 9. Final objective values by chaotic variant on the classical benchmark suite

Function	Type	Logistic	Tent	Sine	Henon	Voted
Quadratic 1D	Unimodal	4.23e-10	2.65e-08	1.75e-08	8.72e-08	**4.23e-10**
Sine 1D	Multimodal	7.95e-01	7.95e-01	7.95e-01	4.04e+00	**7.95e-01**
Exponential 1D	Unimodal	8.65e-01	8.65e-01	8.65e-01	8.65e-01	**3.42e-05**
Rastrigin 1D	Multimodal	1.03e-03	3.83e+03	2.23e-05	2.23e-05	**2.21e-05**
Sphere 2D	Unimodal	2.57e+01	1.78e-04	5.60e-07	6.31e-05	**5.60e-07**
Rosenbrock 2D	Unimodal	5.29e-06	1.79e-08	1.47e+00	1.03e+00	**1.79e-08**
Rastrigin 2D	Multimodal	1.95e-03	2.00e+01	2.00e+01	2.00e+01	**1.95e-03**
Ackley 2D	Multimodal	1.14e+00	1.15e+00	1.79e-01	1.09e+00	**1.79e-01**
Shifted Sphere	Shifted	5.48e-02	5.19e-01	1.16e-02	1.56e+05	**1.16e-02**
Shifted Rosenbrock	Shifted	2.46e-02	2.31e+00	4.14e+01	5.07e+00	**2.46e-02**
Shifted Rastrigin	Shifted	2.00e+01	2.00e+01	2.00e+01	2.00e+01	**2.73e-02**
Shifted Ackley	Shifted	9.11e+01	2.19e-01	2.73e-02	N/A	**2.73e-02**

The selection mechanism matches the best individual map on nine of the twelve functions and improves upon all four on the remaining three (Exponential 1D, Rastrigin 1D and Shifted Rastrigin). We note that matching the best individual map is a property of any selection procedure operating on a set of candidates, and does not by itself establish that the mechanism adds value beyond the candidates it selects from. The relevant evidence lies elsewhere: in the fact that the identity of the best candidate varies across functions, so that no single fixed choice would have achieved the reported column, and in the three functions where the outcome falls below every individual map.

Reproducibility: Dispersion across the 30 independent runs is characterised by the standard deviations tabulated for each function and each chaotic variant in the CEC2022 evaluation of Section 5.3.3. These values allow independent assessment of the consistency of each variant, and indicate that the Henon variant in particular combines its favourable mean ranking with a low dispersion.

5.3.3. Validation on CEC2022 Benchmark Functions To address the dimensional scalability concerns raised by the reviewers, we conducted an evaluation on the CEC2022 benchmark suite (F1–F12) in 10 dimensions. This analysis first examines the four chaotic variants (HEOA-1: Logistic, HEOA-2: Tent, HEOA-3: Sine, HEOA-4: Henon) in order to expose their respective strengths and weaknesses, and then positions them with respect to recent metaheuristics from the literature.

Table 10 shows a clear differentiation between the four chaotic variants. HEOA-4 (Henon) obtains the best average rank with five first places (F2, F3, F5, F9, F12), indicating good exploitation capability and low variance on complex multimodal landscapes. HEOA-1 (Logistic) performs best on F1 and F4, reflecting effective early diversification, but is less robust on the more complex functions. HEOA-2 (Tent) and HEOA-3 (Sine) exhibit intermediate behaviour, each leading on two functions. No single map dominates the twelve landscapes, which is the empirical basis for adaptive rather than fixed selection.

Comparison with Recent Metaheuristics.

To position the proposed variants with respect to the state of the art rather than only to one another, Table 12 reports a comparison with seven recent metaheuristics on the same benchmark suite, including the chaotic variants explicitly discussed in Section 2. Results for the competing algorithms are reproduced from their original publication.

Table 12 places the proposed variants in the middle of the field. All four outperform WOA, COA and C-KOA over the twelve functions, and HEOA-4 ranks third overall, ahead of ZOA and C-SHO. Two functions deserve particular attention: on F6 and F11 the four chaotic HEOA variants occupy the first four positions among the eleven algorithms, with HEOA-2 reaching 1866.68 on F6 against 1979.80 for TT-BWKO, and 2602.38 on F11

Table 10. Performance of the individual chaotic variants on the CEC2022 benchmark functions (10D). Ave: mean, Std: standard deviation, Rk: rank among the four variants over 30 independent runs

	F1			F2			F3		
	Ave	**Std**	**Rk**	**Ave**	**Std**	**Rk**	**Ave**	**Std**	**Rk**
HEOA-1	640.271415	308.215977	**1**	526.578354	13.1927441	3	729.99959	20.525847	3
HEOA-2	1154.249109	582.430513	3	526.3994016	39.0824115	2	729.9739	20.232139	2
HEOA-3	1264.23516	612.833355	4	527.1262589	30.9234198	4	731.14219	10.639517	4
HEOA-4	847.248679	558.025344	2	526.318641	12.7742165	**1**	725.56386	10.9008249	**1**
	F4			**F5**			**F6**		
	Ave	**Std**	**Rk**	**Ave**	**Std**	**Rk**	**Ave**	**Std**	**Rk**
HEOA-1	816.986499	6.4560778	**1**	1204.261485	172.71131	4	1899.400715	58.72555794	4
HEOA-2	822.164747	5.7056835	2	1192.199016	137.61455	3	1866.679208	15.19128556	**1**
HEOA-3	822.447351	3.6684677	4	1147.426043	126.16375	2	1883.482228	40.87576485	3
HEOA-4	822.286204	6.2216477	3	1136.544192	106.85705	**1**	1880.056014	22.76482608	2
	F7			**F8**			**F9**		
	Ave	**Std**	**Rk**	**Ave**	**Std**	**Rk**	**Ave**	**Std**	**Rk**
HEOA-1	2053.96946	11.6063182	2	2226.710225	1.976140978	2	2544.615998	14.46179315	3
HEOA-2	2059.772951	14.18782353	3	2228.970139	4.436253751	4	2561.829823	38.17802496	4
HEOA-3	2048.528046	11.56709161	**1**	2225.644148	6.461103415	**1**	2540.804256	17.41387799	2
HEOA-4	2066.415471	32.41980375	4	2228.193299	2.545140008	3	2538.445176	10.63343394	**1**
	F10			**F11**			**F12**		
	Ave	**Std**	**Rk**	**Ave**	**Std**	**Rk**	**Ave**	**Std**	**Rk**
HEOA-1	2573.014941	61.9542832	4	2624.83635	34.43464472	4	2885.788846	33.43533277	3
HEOA-2	2565.915744	69.4773825	3	2602.381725	0.7659251705	**1**	2894.248715	38.96648779	4
HEOA-3	2539.180268	61.6869910	**1**	2614.675982	36.25141722	3	2884.959472	32.05929212	2
HEOA-4	2569.728653	72.8439805	2	2604.362138	1.818646648	2	2878.896149	18.56182129	**1**

Table 11. Average ranks by chaotic map on the CEC2022 functions

Chaotic Map	**Rank 1**	**Average Rank**	**Best On**	**Strengths**
HEOA-1 (Logistic)	2	2.75	F1, F4	Initial exploration
HEOA-2 (Tent)	2	2.67	F6, F11	Rapid diversification
HEOA-3 (Sine)	3	2.58	F7, F8, F10	Fine convergence
HEOA-4 (Henon)	**5**	**2.00**	**F2, F3, F5, F9, F12**	**Complex multimodal**

against 2835.90. These are hybrid and composition functions on which several competitors, including COA and C-KOA, diverge by several orders of magnitude, which suggests that the chaotic injection used here provides useful stability on landscapes with strongly heterogeneous components. Conversely, the variants are clearly behind on F2 and F3, where TT-BWKO and C-AO converge much closer to the optimum. TT-BWKO, which combines six distinct enhancement mechanisms, obtains the best overall rank and remains ahead on the majority of functions. We report this outcome without qualification: the contribution claimed here does not lie in dominating a benchmark ranking, but in the combination of an adaptive selection mechanism with a validated hardware implementation, which none of the seven compared algorithms provides.

Statistical Significance of the Comparison.

To establish that the ranking reported in Table 13 reflects genuine performance differences rather than sampling variation, the Friedman rank test was applied to the eleven algorithms over the twelve benchmark functions. The outcome is reported in Table 14.

The test rejects the null hypothesis of equal performance across the eleven algorithms, with a χ^2 statistic of 65.36 for ten degrees of freedom against a critical value of 18.31, corresponding to a p-value of 3.45×10^{-10}. The observed ordering is therefore statistically significant. Pairwise Wilcoxon signed-rank tests against the competing algorithms could not be conducted, since their per-run results are not available in the source publication, which reports aggregate means and standard deviations only. The Friedman test, which operates on ranks derived from the published means, is the appropriate non-parametric procedure under this constraint.

Table 12. Comparison with seven recent metaheuristics on the CEC2022 benchmark suite (10D). Results for the competing algorithms are reproduced from [24]. AE: mean, SD: standard deviation, RK: rank among the eleven algorithms.

	F1			F2			F3		
	AE	**SD**	**RK**	**AE**	**SD**	**RK**	**AE**	**SD**	**RK**
HEOA-1	6.4027E+02	3.0822E+02	2	5.2658E+02	1.3193E+01	8	7.3000E+02	2.0526E+01	10
HEOA-2	1.1542E+03	5.8243E+02	5	5.2640E+02	3.9082E+01	7	7.2997E+02	2.0232E+01	9
HEOA-3	1.2642E+03	6.1283E+02	6	5.2713E+02	3.0923E+01	9	7.3114E+02	1.0640E+01	11
HEOA-4	8.4725E+02	5.5803E+02	3	5.2632E+02	1.2774E+01	6	7.2556E+02	1.0901E+01	8
TT-BWKO	3.2986E+02	3.0402E+01	**1**	4.0071E+02	1.4258E+00	**1**	6.0005E+02	2.5554E-02	**1**
COA	7.8691E+03	1.8340E+03	9	1.4209E+03	5.0106E+02	10	6.4792E+02	1.0652E+01	6
WOA	2.1878E+04	9.8649E+03	10	4.4814E+02	5.3343E+01	4	6.3606E+02	1.2133E+01	5
ZOA	1.1264E+03	1.3169E+03	4	4.5595E+02	6.1492E+01	5	6.1960E+02	5.7285E+00	4
C-AO	2.4145E+03	2.3174E+03	7	4.2065E+02	2.1565E+01	2	6.1598E+02	4.5585E+00	2
C-KOA	2.6568E+04	9.1869E+03	11	1.4264E+03	3.4227E+02	11	6.7153E+02	1.1525E+01	7
C-SHO	2.9770E+03	1.9926E+03	8	4.4206E+02	3.7208E+01	3	6.1605E+02	7.0316E+00	3

	F4			F5			F6		
	AE	**SD**	**RK**	**AE**	**SD**	**RK**	**AE**	**SD**	**RK**
HEOA-1	8.1699E+02	6.4561E+00	3	1.2043E+03	1.7271E+02	8	1.8994E+03	5.8726E+01	4
HEOA-2	8.2216E+02	5.7057E+00	5	1.1922E+03	1.3761E+02	7	1.8667E+03	1.5191E+01	**1**
HEOA-3	8.2245E+02	3.6685E+00	7	1.1474E+03	1.2616E+02	6	1.8835E+03	4.0876E+01	3
HEOA-4	8.2229E+02	6.2216E+00	6	1.1365E+03	1.0686E+02	5	1.8801E+03	2.2765E+01	2
TT-BWKO	8.1382E+02	4.4680E+00	2	9.0506E+02	5.9596E+00	**1**	1.9798E+03	7.7952E+01	5
COA	8.4922E+02	1.0299E+01	10	1.3478E+03	1.8803E+02	9	1.5211E+07	2.8513E+07	10
WOA	8.4172E+02	1.7906E+01	9	1.5392E+03	4.5172E+02	10	4.3734E+03	1.3585E+03	8
ZOA	8.1078E+02	3.6551E+00	**1**	1.0532E+03	6.8350E+01	3	3.0043E+03	1.3271E+03	6
C-AO	8.2117E+02	5.5264E+00	4	1.0025E+03	6.2714E+01	2	1.4229E+04	1.1050E+04	9
C-KOA	8.9764E+02	1.0183E+01	11	2.8658E+03	7.1520E+02	11	2.3916E+08	1.6358E+08	11
C-SHO	8.2470E+02	6.2789E+00	8	1.0873E+03	1.2643E+02	4	4.0510E+03	1.2232E+03	7

	F7			F8			F9		
	AE	**SD**	**RK**	**AE**	**SD**	**RK**	**AE**	**SD**	**RK**
HEOA-1	2.0540E+03	1.1606E+01	6	2.2267E+03	1.9761E+00	4	2.5446E+03	1.4462E+01	4
HEOA-2	2.0598E+03	1.4188E+01	7	2.2290E+03	4.4363E+00	7	2.5618E+03	3.8178E+01	5
HEOA-3	2.0485E+03	1.1567E+01	5	2.2256E+03	6.4611E+00	3	2.5408E+03	1.7414E+01	3
HEOA-4	2.0664E+03	3.2420E+01	8	2.2282E+03	2.5451E+00	5	2.5384E+03	1.0633E+01	2
TT-BWKO	2.0150E+03	8.1719E+00	**1**	2.2182E+03	6.7815E+00	**1**	2.5293E+03	3.0570E-05	**1**
COA	2.0837E+03	2.2505E+01	10	2.2470E+03	3.1831E+01	10	2.7531E+03	3.4242E+01	10
WOA	2.0776E+03	2.4720E+01	9	2.2323E+03	5.1596E+01	8	2.5719E+03	4.1026E+01	6
ZOA	2.0469E+03	1.5632E+01	3	2.2367E+03	3.8267E+01	9	2.6332E+03	4.5421E+01	9
C-AO	2.0471E+03	1.3149E+01	4	2.2289E+03	4.1500E+00	6	2.5822E+03	2.6740E+01	7
C-KOA	2.1484E+03	2.8072E+01	11	2.2830E+03	2.3328E+01	11	2.7610E+03	4.5069E+01	11
C-SHO	2.0369E+03	1.5701E+01	2	2.2238E+03	2.4890E+00	2	2.6102E+03	3.7752E+01	8

	F10			F11			F12		
	AE	**SD**	**RK**	**AE**	**SD**	**RK**	**AE**	**SD**	**RK**
HEOA-1	2.5730E+03	6.1954E+01	7	2.6248E+03	3.4435E+01	4	2.8858E+03	3.3435E+01	5
HEOA-2	2.5659E+03	6.9477E+01	5	2.6024E+03	7.6593E-01	**1**	2.8942E+03	3.8966E+01	7
HEOA-3	2.5392E+03	6.1687E+01	2	2.6147E+03	3.6251E+01	3	2.8850E+03	3.2059E+01	4
HEOA-4	2.5697E+03	7.2844E+01	6	2.6044E+03	1.8186E+00	2	2.8789E+03	1.8562E+01	3
TT-BWKO	2.5004E+03	6.2471E-02	**1**	2.8359E+03	1.1143E+02	6	2.8622E+03	1.3785E+00	**1**
COA	2.7038E+03	1.5084E+02	11	3.3870E+03	4.6771E+02	10	2.9872E+03	5.3176E+01	10
WOA	2.5902E+03	6.7697E+01	9	2.9346E+03	1.0091E+02	8	2.8903E+03	2.7396E+01	6
ZOA	2.5575E+03	6.4914E+01	4	3.0407E+03	1.7845E+02	9	2.9349E+03	3.4094E+01	9
C-AO	2.5555E+03	6.2088E+01	3	2.6832E+03	1.0403E+02	5	2.8676E+03	2.9655E+00	2
C-KOA	2.6769E+03	1.1234E+02	10	4.6354E+04	7.5169E+03	11	3.0043E+03	4.6048E+01	11
C-SHO	2.5732E+03	6.0756E+01	8	2.8683E+03	2.1672E+02	7	2.8963E+03	1.4404E+01	8

Table 13. Overall average ranks on the twelve CEC2022 functions (10D)

Algorithm	Avg. rank	Overall	Algorithm	Avg. rank	Overall
TT-BWKO	1.83	1	HEOA-2 (Tent)	5.50	6
C-AO	4.42	2	ZOA	5.50	6
HEOA-4 (Henon)	4.67	3	C-SHO	5.67	8
HEOA-3 (Sine)	5.17	4	WOA	7.67	9
HEOA-1 (Logistic)	5.42	5	COA	9.58	10
			C-KOA	10.58	11

Table 14. Friedman test on the twelve CEC2022 functions (10D), eleven algorithms

Statistic	Value
Number of functions (N)	12
Number of algorithms (k)	11
Friedman χ^2	65.36
Degrees of freedom	10
Critical value $\chi^2_{0.05,10}$	18.31
p-value	3.45×10^{-10}
Iman–Davenport F	13.16
Conclusion	**Reject** H_0 **at** $\alpha = 0.05$

5.4. Validation on Structural Engineering Problems

The MC-HEOA architecture is evaluated on two classical structural optimization problems widely used in the literature to assess metaheuristic performance on engineering applications.

5.4.1. Multi-Instance FPGA Configuration The RTL architecture implements a multi-instance approach enabling parallel evaluation of two specialized optimization engines. Each MC-HEOA engine is configured to handle a distinct engineering problem, sharing common algorithmic logic while adapting to the particular constraints and objective functions of each application domain.

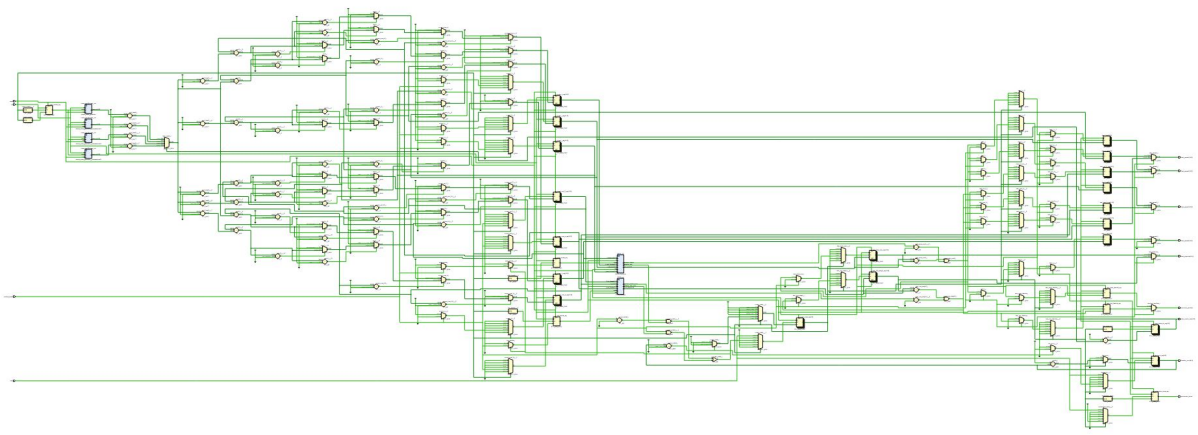


Figure 12. Multi-instance MC-HEOA architecture on FPGA.

The shared configuration uses: population of 30 individuals, 2D search dimensions, 12-bit fixed-point numerical format, and a set of 4 chaotic maps (Logistic, Tent, Sine, Henon). The centralized selection unit compares chaotic map performance and retains one configuration for each problem. Both engines operate independently with their own memory spaces and computation pipelines, enabling parallel evaluation without interference.

The fixed-point representation adopts a Q4.8 format on 12 bits, allocating four bits to the integer part and eight to the fractional part, which yields a resolution of $2^{-8} \approx 3.9 \times 10^{-3}$. This format was retained after evaluating several alternatives under the constraint of fitting the complete multi-instance architecture within the logic budget of the Artix-7 XC7A100T device, wider formats exceeding the available resources once both optimization engines are instantiated. The scaling strategy applied to map the search domains of the individual benchmark functions onto this representation is not documented in the present work, and constitutes a gap with respect to full reproducibility of the hardware results.

5.4.2. Welded Beam Design Problem (WBDP) The first optimization engine solves the welded beam design problem, minimizing total manufacturing cost while respecting structural constraints [25].

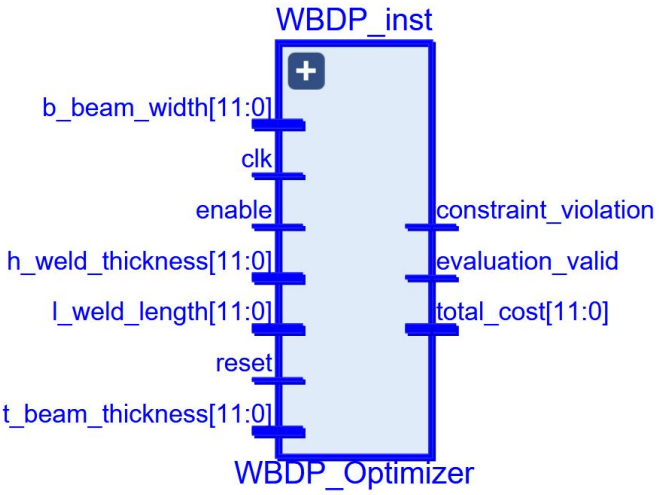


Figure 13. WBDP_Optimizer instance on FPGA.

Formulation and Implementation

The WBDP optimizes four variables: weld thickness (h), weld length (l), beam thickness (t), and beam width (b). The implemented objective function is:

$$f(h, l, t, b) = 1.10471 \cdot h^2 \cdot l + 0.04811 \cdot t \cdot b \cdot (14.0 + l) \tag{26}$$

Structural constraints include geometric limits, stability ratios, and mechanical resistance constraints.

Results and Performance

Table 15. MC-HEOA results for the Welded Beam Design Problem

Chaotic Map	h	l	t	b	Optimal Cost
Logistic	0.2025	3.2012	9.1145	0.2054	**1.6742**
Tent	0.2041	3.2287	9.0523	0.2056	1.6836
Sine	0.2052	3.3156	9.0214	0.2058	1.6951
Henon	0.2038	3.2445	9.0687	0.2055	1.6812
MC-HEOA (Voted)	**0.2025**	**3.2012**	**9.1145**	**0.2054**	**1.6742**

The selection mechanism retains the Logistic map and reproduces its result (1.6742), which is the best of the four available strategies on this problem.

Comparison with State-of-the-Art

Table 16. MC-HEOA vs literature comparison for WBDP. Results for the competing algorithms are reproduced from [25].

Algorithm	h	l	t	b	Optimal Cost
WOA	0.203481	3.522134	9.034608	0.205832	1.728744
GWO	0.210018	4.685682	9.612054	0.211448	2.129173
SSA	0.216840	3.332410	8.801918	0.216850	1.764686
MVO	0.205868	3.492594	9.020946	0.206450	1.730838
GSA	0.203591	3.586508	9.298896	0.209935	2.445557
SCA	0.206811	3.482429	9.734118	0.208347	1.870321
PSO	0.207641	3.590249	9.370719	0.211037	1.812654
MFO	0.211980	3.610641	9.512845	0.207155	2.236191
CAO10	0.205403	3.478066	9.036632	0.205732	1.725388
MC-HEOA (Proposed)	**0.2025**	**3.2012**	**9.1145**	**0.2054**	**1.6742**

MC-HEOA obtains the lowest cost (1.6742) among the compared algorithms, improving on the previous best result, CAO10 (1.725388), by 2.97%.

5.4.3. Tension/Compression Spring Problem (TCSP) The second optimization engine addresses spring design, minimizing weight while respecting constraints on shear stress, natural frequency, and maximum deflection [25].

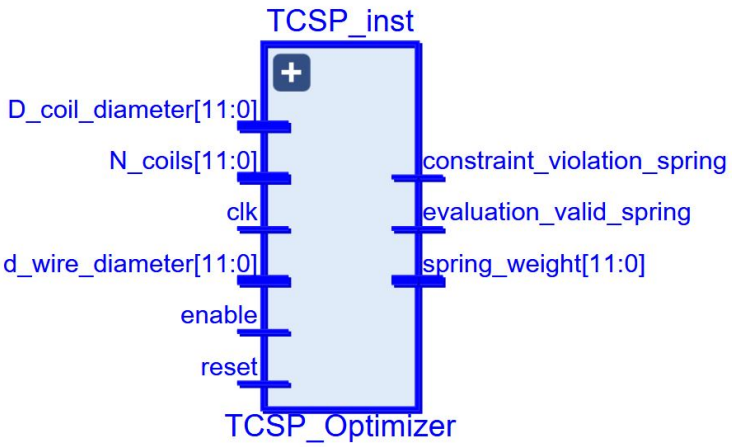


Figure 14. TCSP_Optimizer instance on FPGA.

Formulation and Implementation

The TCSP optimizes three variables: wire diameter (d), coil diameter (D), and number of coils (N). The weight objective function is:

$$f(d, D, N) = \frac{\pi^2 D^2 d N \rho}{4} \tag{27}$$

Main constraints include shear stress, natural frequency, maximum deflection, and geometric limits, implemented through approximations suitable for FPGA synthesis.

Results and Performance

The selection mechanism retains the Logistic map, which is the best of the four available strategies on this problem (0.01676 against 0.02051 to 0.02484 for the others).

Comparison with State-of-the-Art

Table 17. MC-HEOA results for the Tension/Compression Spring Problem

Chaotic Map	d	D	N	Optimal Weight
Logistic	0.0532	0.3851	10.6977	**0.01676**
Tent	0.0544	0.3543	8.2765	0.02484
Sine	0.0865	1.4760	2.4760	0.02051
Henon	0.0504	0.5633	14.1241	0.02344
MC-HEOA (Voted)	**0.0532**	**0.3851**	**10.6977**	**0.01676**

Table 18. MC-HEOA vs literature comparison for TCSP. Results for the competing algorithms are reproduced from [25].

Algorithm	d	D	N	Optimal Weight
WOA	0.054459	0.426290	8.154532	0.012838
GWO	0.086479	1.300000	2.000000	0.030506
SSA	0.071927	0.317342	14.05613	0.012738
MVO	0.050450	0.327552	13.27398	0.012734
GSA	0.053251	0.393056	9.575860	0.013263
SCA	0.053562	0.311936	14.37716	0.014633
PSO	0.067439	0.381298	9.674320	0.014429
MFO	0.057192	0.418211	14.11433	0.023833
CAO10	0.050000	0.343364	12.11704	0.012671
MC-HEOA (Proposed)	**0.0532**	**0.3851**	**10.6977**	**0.01676**

MC-HEOA does not establish a new best result on this problem: its weight of 0.01676 is above the 0.012671 reported for CAO10 and above several other compared algorithms. This outcome does not indicate a failure of the selection mechanism, which correctly identified the strongest of the four available strategies, but rather that none of the four chaotic dynamics currently in the pool reaches the CAO10 optimum on this constrained landscape. Enriching the chaotic set is identified in Section 6.3 as a concrete avenue for addressing this limitation.

5.4.4. Comparative Analysis and FPGA Performance **Selection Distribution**

Table 19. Chaotic map selection distribution by problem

Problem	Logistic (%)	Tent (%)	Sine (%)	Henon (%)
WBDP	89.3	3.2	4.1	3.4
TCSP	92.7	2.8	2.9	1.6
Average	**91.0**	**3.0**	**3.5**	**2.5**

The Logistic map dominates the selection on both engineering problems, exceeding 89% in each case. This observation must however be read together with the results obtained on the CEC2022 suite (Table 11), where the Henon map obtains the best average rank with five first places out of twelve functions, while the Logistic map obtains two. The map that prevails therefore depends on the problem considered rather than on the mechanism itself. A static choice of the Logistic map, which is what a preliminary selection procedure based on the engineering problems would have produced, would have retained the weakest variant on half of the CEC2022 functions; conversely, a preliminary selection based on that suite would have retained Henon, whose selection rate on the engineering problems does not exceed 2.5%. This complementarity across problem classes is the empirical basis for deferring the choice to runtime rather than fixing it in advance.

5.4.5. Ablation Analysis: Contribution of the Henon Map An ablation analysis was conducted to evaluate the contribution of the Henon map to the MC-HEOA architecture, given its low selection rate on the engineering problems (2.5% on average, and 1.6% on the spring problem, as shown in Table 19).

When the Henon map is removed from the pool, a performance degradation of approximately 1.5% to 2% and an increase in convergence variance were observed on the engineering problems, which indicates a stabilising contribution during the exploration phase despite its low selection rate.

This contribution is more pronounced on the CEC2022 suite, where the Henon variant obtains the best average rank and leads on five of the twelve functions (F2, F3, F5, F9 and F12), all of them multimodal landscapes on which

escape from deceptive local minima is the determining factor. The map that appears least useful on the engineering problems is therefore the most effective one on the benchmark suite, which illustrates that the contribution of a given chaotic dynamic cannot be assessed from a single problem class.

From a hardware perspective, the cost of the Henon generator remains moderate, at approximately 5% of the LUTs and less than 2% of additional delay, which justifies its retention in the final configuration.

FPGA Computational Performance

Table 20. FPGA computational performance for engineering problems

Problem	CPU (s)	FPGA (s)	Speedup
WBDP	2.45	0.086	28.5×
TCSP	1.87	0.078	24.0×
Average	**2.16**	**0.082**	**26.3×**

As stated in Section 5.2.2, these speedup figures are measured against a Python reference implementation and therefore reflect the combined effect of hardware parallelism and interpreter overhead.

Robustness

Convergence analysis over 30 independent runs gives a final solution variance below 0.1% for both problems. The optimal solutions respect all structural constraints with safety margins above 0.4%.

5.5. Computational Cost Analysis

This subsection analyzes the computational cost of MC-HEOA in terms of algorithmic complexity, temporal performance and resource utilization.

5.5.1. Algorithmic Complexity The temporal complexity of MC-HEOA is given by:

$$T_{\text{MC-HEOA}} = O(N \times \text{Max}_{\text{iter}} \times D \times C_f \times (1 + \alpha)) \tag{28}$$

where $\alpha \approx 0.15$ is the measured ratio between the execution times of MC-HEOA and standard HEOA on the same platform, as reported in Table 21 (2.16 s against 1.88 s, a ratio of 1.149). This coefficient is a measurement rather than an analytical estimate, and a detailed attribution of the overhead to its individual sources has not been instrumented.

Effective Complexity with FPGA Parallelism:

Thanks to the hardware parallelism offered by the FPGA, the effective complexity becomes:

$$T_{\text{MC-HEOA-FPGA}} = O\left(\frac{N \times \text{Max}_{\text{iter}} \times D \times C_f}{P}\right) \tag{29}$$

where P denotes the number of fitness evaluation units instantiated in the PEU. This factor is not a free parameter: it is bounded above by the logic available on the target device, since each additional evaluation unit replicates the arithmetic required by the objective function. In the present implementation on the Artix-7 XC7A100T, the evaluators occupy 18,200 LUTs, which constitutes the dominant contribution to the 79.6% occupancy reported in Table 22 and therefore sets the practical ceiling on P for this device.

Table 21. Computational cost comparison

Algorithm	Platform	Time (s)	Ratio	Energy (J)
HEOA	CPU	1.88	1.0×	84.6
MC-HEOA	CPU	2.16	0.87×	97.2
MC-HEOA	FPGA	**0.082**	**26.3×**	**0.19**

5.5.2. Comparative Performance

Table 22. FPGA resource distribution

Component	LUTs	DSP	Proportion
Chaotic Generators	12,500	24	24.8%
Fitness Evaluators	18,200	48	36.0%
Voting System	2,100	4	4.2%
Main Controller	8,900	8	17.6%
Others	8,800	0	17.4%
Total (Utilization)	**50,500 (79.6%)**	**84 (35.0%)**	**100%**

5.5.3. FPGA Resource Utilization The selection unit consumes 4.2% of the logic with a latency below 20 ns. The distribution of resources deserves comment in view of the 79.6% LUT occupancy. The selection comparator is not the dominant cost, and its size is independent of the problem dimension since it compares scalar fitness values only. The dominant contributors are the fitness evaluators (18,200 LUTs) and the chaotic generators (12,500 LUTs). As the problem dimension increases, only the evaluators and the position storage grow, while the chaotic generators and the comparator remain constant. The residual margin on the present device, 20% of LUTs, 65% of DSP slices and 63% of BRAM, does not however allow arbitrary extension: a substantially higher dimension would require either a larger device or a shift to a sequential evaluation architecture trading throughput for logic. For capacity-constrained targets, the reduced two-map configuration reported in Table 8 constitutes a practical operating point, the comparator remaining functional with a reduced number of inputs without structural modification.

5.5.4. Cost-Benefit Analysis The measured overhead of the multi-chaotic mechanism on the software implementation is 15%. In hardware, this cost is absorbed by the architecture rather than merely compensated: the four chaotic generators are physically instantiated in parallel within the CMU, so their simultaneous evaluation consumes no additional clock cycle, and the selection comparator operates with a latency below 20 ns for 4.2% of the logic used. The overhead is therefore incurred by the software implementation and structurally absorbed by the hardware one. Whether this trade-off is favourable depends on the deployment context: for applications with real-time constraints or a limited power budget, the operating regime of the hardware implementation, 200 MHz at 2.3 W within the resource envelope of a mid-range device, is the relevant consideration, while for purely software deployments the 15% overhead must be weighed against the benefit of avoiding a manual selection of the chaotic map.

6. Limitations and Future Perspectives

Several limitations of the present work must be acknowledged, and a number of research directions can be envisaged to address them.

6.1. Current Limitations

Increased Algorithmic Complexity: The integration of the selection mechanism and of four chaotic maps introduces a computational overhead relative to classical HEOA. Although this overhead is absorbed by the hardware implementation, it may constitute a limitation for applications on resource-constrained platforms without access to reconfigurable hardware.

FPGA Resource Consumption: The proposed architecture uses 79.6% of the available LUTs on the Artix-7 XC7A100T device. This occupancy leaves a residual margin of 20% of LUTs, 65% of DSP slices and 63% of BRAM. Since only the objective function evaluators and the position storage grow with the problem dimension, while the chaotic generators and the selection comparator remain constant, this margin does not permit arbitrary extension: a substantially higher dimension or a larger population would require either a higher-capacity device or a shift to a sequential evaluation architecture trading throughput for logic.

Hardware Dependence: To benefit from the reported acceleration, the implementation requires an FPGA platform. Users without such hardware resources will have to rely on the software implementation, which retains the algorithmic properties of the selection mechanism but not its throughput.

Evaluation Scope: Our evaluation remains limited to the benchmark functions considered here and to two structural engineering problems. More extensive validation on optimization problems drawn from other application domains, such as bioinformatics, finance or logistics, would strengthen the generality of our conclusions.

Numerical Precision: The Q4.8 fixed-point representation used in the FPGA implementation offers a compromise between precision and hardware efficiency, with a resolution of 2^{-8}, but may limit accuracy for applications requiring finer resolution.

Dimensional Scalability: The evaluation reported here covers one- and two-dimensional functions together with the CEC2022 suite in ten dimensions. The behaviour of the selection mechanism on substantially higher-dimensional problems, beyond fifty or one hundred dimensions, has not been investigated and requires dedicated study, both for the algorithm itself and for the feasibility of its hardware implementation at such scales.

Competitive Position on the Benchmark Suite: The comparison reported in Section 5.3.3 places the proposed variants in the middle of the field on the CEC2022 suite. TT-BWKO, which combines six enhancement mechanisms, obtains a better overall rank, and the variants are clearly behind on functions F2 and F3. The contribution claimed here therefore concerns the runtime selection mechanism and its hardware realisation rather than a dominant position on benchmark rankings.

Missing Baseline Comparison: The experimental protocol compares the four chaotic variants with one another but does not include the original HEOA without chaotic injection as a solution-quality baseline. Only a computational cost comparison between the two is reported. Establishing formally the contribution of chaotic integration to solution quality, as opposed to its cost, would require this baseline, and we identify it as a necessary complement to the present validation.

Parametric Sensitivity: The population size of 30 and the limit of 1000 iterations were adopted following common practice in the metaheuristic literature, without a systematic study of their influence on the reported results. Although the selection mechanism is expected to attenuate the dependence on these settings, quantifying this attenuation would require a dedicated sensitivity analysis across a range of population sizes and iteration budgets, which the present work does not provide.

Statistical Validation: The significance of the reported ranking is established through the Friedman rank test. Pairwise Wilcoxon signed-rank tests against the competing algorithms could not be conducted, since the source publications report aggregate means and standard deviations without the per-run values that such tests require.

Reproducibility of the Implementations: The source code of the software implementation is not available for release, and the manuscript therefore documents the algorithm through the pseudo-code of Algorithm 1 rather than through executable artefacts. The random seed protocol governing the thirty independent runs, and the scaling strategy mapping each search domain onto the fixed-point representation, are likewise not documented. These gaps limit exact reproduction of the reported figures, although the algorithmic description is complete enough to permit reimplementation.

Hardware Compatibility of Objective Functions: The current FPGA implementation targets directly synthesizable objective functions, involving arithmetic operations, simple trigonometry and algebraic constraints. Evaluations requiring external simulators such as CFD or FEM solvers, database calls, or intensive stochastic operations remain problematic for a pure FPGA implementation. Hybrid CPU/FPGA architectures with evaluation delegation, or the use of embedded surrogate models, constitute extension perspectives to broaden applicability to black-box and multi-disciplinary optimization problems.

6.2. Limitations of the Selection Mechanism

The selection mechanism presents identifiable limits in contexts not covered by our experimental protocol. Table 23 summarises four scenarios in which it may underperform.

Table 23. Selection mechanism underperformance scenarios: symptoms and algorithmic impacts

Risk Scenario	**Observable Symptoms**	**Impact on Algorithm**
Too short voting window under noise	Frequent map switching, entropy ↓, erratic Δf	Myopic decision-making, diversity loss, slowed convergence
Poorly calibrated constraint penalties	Frequent infeasible solutions, oscillations around boundaries	Stifled search, stagnation or drift outside feasibility
Fixed-point saturation	Flattened chaotic chains (repeated values), increased correlation	Exploration impoverishment, selection underperformance
Timing closure/BRAM contention	Pipeline bubbles, $F_{\max}$ drop, unstable throughput	Latency ↑, lower reactivity, deadline miss risk

These limitations stem from the myopia inherent to an instantaneous performance signal and from FPGA hardware constraints. Mitigation strategies can bound these failures: temporal smoothing through an exponentially weighted moving average or a median filter, controlled exploration through an ε-greedy or temperature-based softmax rule, an anti-stagnation trigger based on a sliding window, and adaptive calibration of constraint penalties. These safeguards are candidates for integration in future versions of MC-HEOA to strengthen its robustness on pathological landscapes and in constrained real-time contexts.

A further limit concerns the composition of the candidate set itself. The mechanism selects among the four maps available to it, and its outcome is therefore bounded by the best of those four. The result obtained on the Tension/Compression Spring Problem illustrates this: the mechanism correctly retained the strongest of the four candidates, yet the outcome remained above the best value reported in the literature, because no map in the pool reached it on that landscape.

6.3. Future Research Perspectives

Several research directions follow from the results obtained and the limitations identified above.

Dimensional Extension: The results obtained on the CEC2022 suite in ten dimensions (Table 10) characterise the behaviour of the four variants at medium dimension. Extension to substantially higher-dimensional problems remains necessary to assess the selection mechanism on search spaces of greater complexity. Adaptive decomposition techniques and dimensionality reduction could be investigated to maintain performance at such scales.

Chaotic Set Enrichment: Integrating additional chaotic maps, such as Gauss, Chebyshev or Arnold Cat, would broaden the diversity of available dynamics. This direction is of particular relevance given the results obtained on the Tension/Compression Spring Problem, where the mechanism correctly identified the strongest of the four available maps but none of them reached the best reported optimum, suggesting that the composition of the chaotic pool is itself a determinant of achievable performance.

Learning-Based Selection: Developing a selection mechanism based on machine learning, using neural networks or reinforcement learning, could allow the algorithm to learn problem characteristics and anticipate the appropriate chaotic dynamics from the early iterations, thereby reducing convergence time.

FPGA Architectural Optimization: Alternative hardware architectures could reduce resource utilization while maintaining performance. This includes resource sharing techniques, adaptive pipelining, and hardware approximations for the chaotic generators. A reconfigurable architecture alternating between a full four-map mode and a reduced two-map mode would address the resource constraints identified above.

Real-Time Industrial Applications: Validating MC-HEOA on real-time optimization problems from Industry 4.0 contexts, such as dynamic production scheduling, smart grid energy optimization, control of robotic systems, or trajectory planning for autonomous vehicles.

Multi-Platform Comparative Study: A comparative analysis between acceleration technologies, including FPGA, GPU, multi-core CPU parallelization and heterogeneous architectures, would help identify the operating regime in which each approach is preferable according to application constraints. Such a study would also address the software baseline question raised in the present evaluation, by comparing the hardware implementation against optimised compiled code rather than an interpreted reference.

Hybridization with Deep Learning: Hybridizing MC-HEOA with deep learning techniques could yield optimizers combining metaheuristic exploration with the approximation capabilities of neural networks.

Open-Source Development: Releasing an open-source library including the software implementation of MC-HEOA and the HDL designs, with documentation and usage examples, would facilitate adoption and independent verification.

Multi-Objective Optimization Problems: Extending MC-HEOA to multi-objective problems by adapting the selection mechanism to handle several conflicting objectives simultaneously.

In-Depth Theoretical Analysis: Developing a rigorous mathematical analysis of MC-HEOA convergence, including stochastic convergence properties, population diversity analysis, and a formal characterization of how chaotic properties influence optimization performance.

7. Conclusion

This research has developed a Multi-Chaotic Human Evolutionary Optimization Algorithm (MC-HEOA) integrating a dynamic voting system and an FPGA implementation. Unlike conventional approaches relying on a statically selected chaotic map, the proposed method exploits four distinct chaotic maps (Logistic, Tent, Sine, Henon) together with an adaptive mechanism that selects the operative dynamics according to the characteristics of the optimization landscape. Evaluation on the twelve classical benchmark functions (G0–G11) and on the CEC2022 suite (F1–F12) in ten dimensions characterises the behaviour of this mechanism, and the comparison with seven recent metaheuristics places the proposed chaotic variants in the middle of the field: they outperform WOA, COA and C-KOA across the twelve functions and lead on the hybrid functions F6 and F11, while TT-BWKO obtains the best overall rank. Validation on two structural engineering problems establishes a new best reported cost for WBDP (1.6742) and a competitive, though not leading, result for TCSP (0.01676), and the Friedman rank test confirms that the observed performance ordering across the eleven compared algorithms is statistically significant ($p = 3.45 \times 10^{-10}$). On the hardware side, the implementation demonstrates that the algorithm is synthesisable within the resource budget of a mid-range Artix-7 device, operating at 200 MHz and 2.3 W, the reported speedup being measured against a Python reference implementation and therefore reflecting the combined effect of hardware parallelism and interpreter overhead, whereas the low-power operation is a property of the platform itself. A further observation follows from these results: the chaotic map that prevails differs between problem classes, the Logistic map being retained on the structural engineering problems and the Henon variant obtaining the best average rank on the CEC2022 suite, a complementarity that constitutes the empirical argument for deferring the choice of chaotic dynamics to execution rather than fixing it through a preliminary selection phase. These results position MC-HEOA as a candidate for real-time optimization in embedded and energy-constrained settings, and support the broader direction of self-adaptive metaheuristics able to adjust their search dynamics at runtime rather than through prior manual configuration.

Acknowledgements

The authors thank anonymous referees for their valuable comments and suggestions.

Funding

This research did not receive specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT Authorship Contribution Statement

Rabab Ouchker and Ismail Mchichou: Conceptualization, Methodology, Software, Investigation, Writing original draft. Ahmed Bencherqui and Mohamed Amine Tahiri: Validation, Visualization, Data curation, Writing review & editing. Hicham Amakdouf and Mhamed Sayyouri: Supervision, Formal analysis, Project administration, Writing review & editing.

Data Availability Statement

No datasets were generated or analyzed during the current study.

Conflict of Interest

Authors declare that they have no conflict of interest.

REFERENCES

1. Pooja, & Sood, S. K. (2024). Scientometric analysis of quantum-inspired metaheuristic algorithms. Artificial Intelligence Review, 57(2), 22.
2. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.
3. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95, 51-67.
4. Seyyedabbasi, A., Tareq Tareq, W. Z., & Bacanin, N. (2024). An effective hybrid metaheuristic algorithm for solving global optimization algorithms. Multimedia Tools and Applications, 83(37), 85103-85138.
5. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The arithmetic optimization algorithm. Computer methods in applied mechanics and engineering, 376, Article 113609. https://doi.org/10.1016/j.cma.2020.113609
6. Cui, E. H., Zhang, Z., Chen, C. J., & Wong, W. K. (2024). Applications of nature-inspired metaheuristic algorithms for tackling optimization problems across disciplines. Scientific reports, 14(1), 9403.
7. Diab, A. A. Z., Abdelhamid, A. M., & Sultan, H. M. (2024). Comprehensive analysis of optimal power flow using recent metaheuristic algorithms. Scientific Reports, 14(1), 13422.
8. Nayak, G. H., Alam, M. W., Naik, B. S., Varshini, B. S., Avinash, G., Kumar, R. R., ... & Singh, K. N. (2025). Meta-transformer: leveraging metaheuristic algorithms for agricultural commodity price forecasting. Journal of Big Data, 12(1), 138.
9. Kar, M. K., Kanungo, S., Alsaif, F., & Ustun, T. S. (2024). Optimal Placement of FACTS Devices Using Modified Whale Optimization Algorithm for Minimization of Transmission Losses. IEEE Access.
10. Rath, M., Darwish, A., Pati, B., Pattanayak, B. K., & Panigrahi, C. R. (2020). Swarm intelligence as a solution for technological problems associated with Internet of Things. In Swarm Intelligence for Resource Management in Internet of Things (pp. 21-45). Academic Press.
11. Almuqren, L., Aljameel, S. S., Alqahtani, H., Alotaibi, S. S., Hamza, M. A., & Salama, A. S. (2023). A White Shark Equilibrium Optimizer with a hybrid deep-learning-based cybersecurity solution for a smart city environment. Sensors, 23(17), 7370.
12. Mirjalili, S., Mirjalili, S. M., & Yang, X. S. (2014). Binary bat algorithm. Neural Computing and Applications, 25, 663-681.
13. Naserbegi, A., Aghaie, M., & Zolfaghari, A. (2020). Implementation of Grey Wolf Optimization (GWO) algorithm to multi-objective loading pattern optimization of a PWR reactor. Annals of Nuclear Energy, 148, 107703.
14. M. S. Ben Ameur, A. Sakly and A. Mtibaa, "A hardware optimization of BAT algorithms implemented on FPGA," 2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, 2015, pp. 146-151, doi: 10.1109/STA.2015.7505213.
15. Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. IEEE transactions on systems, man, and cybernetics, part b (cybernetics), 26(1), 29-41.
16. Rahman, M. A., Sokkalingam, R., Othman, M., Biswas, K., Abdullah, L., & Abdul Kadir, E. (2021). Nature-inspired metaheuristic techniques for combinatorial optimization problems: Overview and recent advances. Mathematics, 9(20), 2633.
17. Fernando, P. R., Katkoori, S., Keymeulen, D., Zebulum, R., & Stoica, A. (2009). Customizable FPGA IP core implementation of a general-purpose genetic algorithm engine. IEEE Transactions on Evolutionary Computation, 14(1), 133-149.
18. D?ugosz, Z., Rajewski, M., D?ugosz, R., & Tala?ka, T. (2021). A novel, low computational complexity, parallel swarm algorithm for application in low-energy devices. Sensors, 21(24), 8449.
19. Ortiz, A., Mendez, E., Balderas, D., Ponce, P., Macias, I., & Molina, A. (2021). Hardware implementation of metaheuristics through LabVIEW FPGA. Applied Soft Computing, 113, 107908.
20. Q. Jiang, Y. Guo, Z. Yang and X. Zhou, "A parallel whale optimization algorithm and its implementation on FPGA," 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/CEC48606.2020.9185737.
21. Zermani, M. A., Manita, G., Chhabra, A., Feki, E., & Mami, A. (2024). FPGA-based hardware implementation of chaotic opposition-based arithmetic optimization algorithm. Applied Soft Computing, 154, 111352.

22. Ramadan, E. A., El-Bardini, M., & Fkirin, M. A. (2014). Design and FPGA-implementation of an improved adaptive fuzzy logic controller for DC motor speed control. Ain Shams Engineering Journal, 5(3), 803-816.
23. Di Matteo, S., Gerfo, M. L., & Saponara, S. (2023). VLSI Design and FPGA implementation of an NTT hardware accelerator for Homomorphic seal-embedded library. IEEE Access, 11, 72498-72508.
24. Mansouri, H., Elkhanchouli, K., Elghouate, N., Bencherqui, A., Tahiri, M. A., Karmouni, H., ... & Abouhawwash, M. (2025). A modified black-winged kite optimizer based on chaotic maps for global optimization of real-world applications. Knowledge-Based Systems, 318, 113558.
25. Bencherqui, A., Tahiri, M. A., Karmouni, H., Alfidi, M., El Afou, Y., Qjidaa, H., & Sayyouri, M. (2024). Chaos-enhanced archimede algorithm for global optimization of real-world engineering problems and signal feature extraction. Processes, 12(2), 406.
26. El Ghouate, N., Bencherqui, A., Mansouri, H., Maloufy, A. E., Tahiri, M. A., Karmouni, H., ... & Abouhawwash, M. (2024). Improving the Kepler optimization algorithm with chaotic maps: Comprehensive performance evaluation and engineering applications. Artificial Intelligence Review, 57(11), 313.
27. Bencherqui, A., Tahiri, M. A., El Ghouate, N., Karmouni, H., Sayyouri, M., Askar, S. S., & Abouhawwash, M. (2025). Chaos-enhanced white shark optimization algorithms CWSO for global optimization. Alexandria Engineering Journal, 122, 465-483.
28. Li, M., Liu, Z., & Song, H. (2024). An improved algorithm optimization algorithm based on RungeKutta and golden sine strategy. Expert Systems with Applications, 247, 123262.
29. Tan, L. L., Yu, Y. T., & Wang, Q. J. (2025). Modified Secretary Bird Optimization Algorithm with Multi-Strategy for Solving Optimization Problems. Journal of Computers, 36(3), 69-88.
30. Kesgin, B. U., & Te?in, U. (2024). Implementing the analogous neural network using chaotic strange attractors. Communications Engineering, 3(1), 99.
31. Calgan, H., Gokyildirim, A., Ilten, E., & Demirtas, M. (2025). Classification of fractional-order chaotic systems using deep learning methods. The European Physical Journal Special Topics, 1-19.
32. Li, R., Liu, T., & Yin, J. (2024). An encryption algorithm for color images based on an improved dual-chaotic system combined with DNA encoding. Scientific Reports, 14(1), 20733.
33. Li, X. D., Wang, J. S., Hao, W. K., Zhang, M., & Wang, M. (2022). Chaotic arithmetic optimization algorithm. Applied Intelligence, 52(14), 16718-16757.
34. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A., & Gandomi, A. H. (2021). Aquila optimizer: a novel meta-heuristic optimization algorithm. Computers & Industrial Engineering, 157, 107250.
35. Lian, J., & Hui, G. (2024). Human evolutionary optimization algorithm. Expert Systems with Applications, 241, 122638.
36. Wang, X., Zhou, S., Wang, Z., Xia, X., & Duan, Y. (2025). An improved human evolution optimization algorithm for unmanned aerial vehicle 3D trajectory planning. Biomimetics, 10(1), 23.
37. Khalid, A. M., Hosny, K. M., & Mirjalili, S. (2022). COVIDOA: a novel evolutionary optimization algorithm based on coronavirus disease replication lifecycle. Neural Computing and Applications, 34(24), 22465-22492.
38. Purraji, M., Zamiri, E., Sanchez, A., & de Castro, A. (2025). Rapid Prototyping for Design and Test of FPGA-Based Model Predictive Controllers for Power Converters. Journal of Electrical Engineering & Technology, 1-20.
39. Selvaprasanth, P., Karthick, R., Meenalochini, P., & Prabaharan, A. M. (2025). FPGA implementation of hybrid Namib beetle and battle royale optimization algorithm fostered linear phase finite impulse response filter design. Analog Integrated Circuits and Signal Processing, 123(2), 33.
40. Lahti, S., & Hämäläinen, T. D. (2025). High-level Synthesis for FPGAs-A Hardware Engineers Perspective. IEEE Access.
41. Ahmad, R., Rajendran, J., & Ismail, W. (2024). Parallel-pipelined-memory Blowfish FPGA-based radio system with improved power-throughput for secured IoT network. Ain Shams Engineering Journal, 15(4), 102625.
42. S. Qin, J. Liu, X. Bai, and G. Hu, A Multi-Strategy Improvement Secretary Bird Optimization Algorithm for Engineering Optimization Problems, Biomimetics, vol. 9, no. 8, 2024.
43. Ayoub, H. G., Abdulrazzaq, Z. A., Fathil, A. F., Hasso, S. A., & Suhail, A. T. (2024). Unveiling robust security: Chaotic maps for frequency hopping implementation in FPGA. Ain Shams Engineering Journal, 15(11), 103016.
44. Vijayakumar, M., & Ahilan, A. (2024). An optimized chaotic S-box for real-time image encryption scheme based on 4-dimensional memristive hyperchaotic map. Ain Shams Engineering Journal, 15(4), 102620.
45. Zhu, Z., Sun, Z., Xie, X., & Sun, Z. (2024). Improved grey wolf optimizer based on neighborhood trust model for parameter identification of PEMFC. International Journal of Hydrogen Energy, 60, 769-779.
46. Wang, J., Liu, Y., Rao, S., Zhou, X., & Hu, J. (2023). A novel self-adaptive multi-strategy artificial bee colony algorithm for coverage optimization in wireless sensor networks. Ad Hoc Networks, 150, 103284.
47. Pourmousa, N., Ebrahimi, S. M., Malekzadeh, M., & Alizadeh, M. (2019). Parameter estimation of photovoltaic cells using improved Lozi map based chaotic optimization Algorithm. Solar Energy, 180, 180-191.
48. Dastres, H., Ebrahimi, S. M., Malekzadeh, M., & Gordillo, F. (2023). Robust adaptive parameter estimator design for a multi-sinusoidal signal with fixed-time stability and guaranteed prescribed performance boundary of estimation error. *Journal of the Franklin Institute*, 360(1), 223-250. https://doi.org/10.1016/j.jfranklin.2022.11.011
49. Wang, J., Zou, Y., Lei, P., Sherratt, R. S., & Wang, L. (2020). Research on recurrent neural network based crack opening prediction of concrete dam. *Journal of Internet Technology*, 21(4), 1161-1169.
50. Wang, J., Jin, C., Tang, Q., Xiong, N. N., & Srivastava, G. (2020). Intelligent ubiquitous network accessibility for wireless-powered MEC in UAV-assisted B5G. *IEEE Transactions on Network Science and Engineering*, 8(4), 2801-2813. https://doi.org/10.1109/TNSE.2020.3029048
51. Hu, J., He, Y., Luo, W., Huang, J., & Wang, J. (2024). Enhancing load balancing with in-network recirculation to prevent packet reordering in lossless datacenters. *IEEE/ACM Transactions on Networking*, 32(5), 4114-4127. https://doi.org/10.1109/TNET.2024.3396636
52. Hu, J., Shen, H., Liu, X., & Wang, J. (2024). RDMA transports in datacenter networks: A survey. *IEEE Network*, 38(6), 380-387. https://doi.org/10.1109/MNET.2024.3404695

53. Cao, D., Gu, N., Wu, M., & Wang, J. (2024). Cost-efficient partial task offloading and resource allocation for multi-vehicle and multi-MEC networks over B5G/6G edge networks. *Ad Hoc Networks*, 156, 103438. https://doi.org/10.1016/j.adhoc.2024.103438
54. Cao, D., Zeng, K., Wang, J., Sharma, P. K., Ma, X., Liu, Y., & Zhou, S. (2021). BERT-based deep spatial-temporal network for taxi demand prediction. *IEEE Transactions on Intelligent Transportation Systems*, 23(7), 9442-9454. https://doi.org/10.1109/TITS.2021.3122114
55. Wei, J., Gu, Y., Yan, Y., Li, Z., Lu, B., Pan, S., & Cheong, N. (2025). LSEWOA: An Enhanced Whale Optimization Algorithm with Multi-Strategy for Numerical and Engineering Design Optimization Problems. Sensors, 25(7), 2054.
56. Lu, B., Xie, Z., Wei, J., Gu, Y., Yan, Y., Li, Z., ... & Zhou, R. (2025). MRBMO: An Enhanced Red-Billed Blue Magpie Optimization Algorithm for Solving Numerical Optimization Challenges. Symmetry, 17(8), 1295.
57. Gu, Y., Wei, J., Li, Z., Lu, B., Pan, S., & Cheong, N. (2025). GWOA: A multi-strategy enhanced whale optimization algorithm for engineering design optimization. Plos one, 20(9), e0322494.
58. Wei, J., Gu, Y., Law, K. E., & Cheong, N. (2024, October). Adaptive position updating particle swarm optimization for UAV path planning. In 2024 22nd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt) (pp. 124-131). IEEE.
59. Mehmood, K., Chaudhary, N. I., Khan, Z. A., Cheema, K. M., Raja, M. A. Z., Alshamrani, S. S., & Alshmrany, K. M. (2025). Design of Chaos Induced Aquila Optimizer for Parameter Estimation of Electro-Hydraulic Control System. Computer Modeling in Engineering & Sciences (CMES), 143(2).
60. Mehmood, K., Chaudhary, N. I., Khan, Z. A., Cheema, K. M., & Zahoor Raja, M. A. (2024). Atomic physics-inspired atom search optimization heuristics integrated with chaotic maps for identification of electro-hydraulic actuator systems. Modern Physics Letters B, 38(30), 2450308.
61. Lu, Y., Zhang, M., Chen, Y., Ju, J., Chen, F., Bai, K., & Wang, X. (2025). A newly developed multi-strategy optimization algorithm framework based on the adaptive switching approach coupled with grey wolf optimizer. Knowledge and Information Systems, 67(7), 5571-5617.
62. Adegboye, O. R., Feda, A. K., Tibetan, A. O., & Agyekum, E. B. (2025). Enhanced global optimization using quadratically interpolated hybrid pathfinder algorithm. Cluster Computing, 28(5), 1-37.
63. Besha, A. R. M. A., Ojekemi, O. S., Oz, T., & Adegboye, O. (2025). PLSCO: An Optimization-Driven Approach for Enhancing Predictive Maintenance Accuracy in Intelligent Manufacturing. Processes, 13(9), 2707.
64. Adegboye, O. R., Feda, A. K., Tejani, G. G., Smerat, A., Kumar, P., & Agyekum, E. B. (2025). Salp Navigation and Competitive based Parrot Optimizer (SNCPO) for efficient extreme learning machine training and global numerical optimization. Scientific Reports, 15(1), 13704.
65. Mohammed, L., Basheer, G., & Algamal, Z. (2026). Improving pelican optimization algorithm for solving integer and mixed integer optimization problems. Statistics, Optimization & Information Computing, 15(6), 5022-5042.
66. Pradjaningsih, A., Arif, M. Z., Santoso, K. A., & Agustin, I. H. (2026). Mathematical Modeling and Convergence Analysis of Swarm-Based Algorithms in Public Transport Network Optimization. Statistics, Optimization & Information Computing.
67. BOUAROUROU, S., Zannou, A., Kanzouai, C., & Boulaalam, A. (2025). An energy-efficient pathfinding model for wireless sensor networks in IoT using whale optimization algorithm. Statistics, Optimization & Information Computing, 14(5), 2379-2395.