IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 September 2022, accepted 16 October 2022, date of publication 21 October 2022, date of current version 28 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3216321

RESEARCH ARTICLE

A New Algorithm Inspired on Reversible Elementary Cellular Automata for Global Optimization

JUAN CARLOS SECK-TUOH-MORA[1], OMAR LOPEZ-ARIAS[1], NORBERTO HERNANDEZ-ROMERO[1], GENARO J. MARTÍNEZ[2,3], AND VALERIA VOLPI-LEON[1]

[1]Área Académica de Ingeniería y Arquitectura, Instituto de Ciencias Básicas e Ingeniería, Universidad Autónoma del Estado de Hidalgo, Pachuca, Hidalgo 42184, Mexico
[2]Artificial Life Robotics Laboratory, Escuela Superior de Computo, Instituto Politecnico Nacional, Mexico City 07340, Mexico
[3]Unconventional Computing Laboratory, University of the West of England, BS16 1QY Bristol, U.K.

Corresponding author: Juan Carlos Seck-Tuoh-Mora (jseck@uaeh.edu.mx)

This work was supported in part by the National Council for Science and Technology [Consejo Nacional de Ciencia y Tecnología (CONACYT)] under Project F003/320109. The work of Omar Lopez-Arias was supported by the CONACYT under Grant 850822.

ABSTRACT This work presents a new global optimization algorithm of functions inspired by the dynamic behavior of reversible cellular automata, denominated Reversible Elementary Cellular Automata Algorithm (RECAA). This algorithm adapts the reversible evolution rules in elementary cellular automata (in one dimension and only with two states) to work with vectors of real values to realize optimization tasks. The originality of RECAA lies in adapting the dynamic of the reversible elementary cellular automata to perform exploration and exploitation actions in the optimization process. This work shows that diversity in cellular automata behaviors (in this case, reversibility) is useful to define new metaheuristics to solve optimization problems. The algorithm is compared with 15 recently published metaheuristics that recognized for their good performance, using 50 test functions in 30, 500, and with a fixed number of dimensions, and the CEC 2022 benchmark suit. Additionally, it is shown that RECAA has been applied in 3 engineering problems. In all the experiments, RECAA obtained satisfactory results. RECAA was implemented in MATLAB, and its source code can be consulted in GitHub. `https://github.com/juanseck/RECAA`

INDEX TERMS Engineering applications, global optimization, metaheuristics, reversible cellular automata.

I. INTRODUCTION

In these times, the problems in engineering involve an increasing number of variables, with a higher nonlinear connection between them which makes their optimization more challenging. This feature is clearly observed in various application fields, such as mechanical design, chemistry engineering, power electronics, or energy generation systems problems [1].

In problems that, by their nature, are highly nonlinear, and on many occasions, non-differentiable, non-separable, and multimodal, the classic optimization algorithms based on gradient methods are not effective since they are trapped in local minima or lack the necessary conditions to converge properly [2]. In this situation, a great diversity of new metaheuristic algorithms have emerged which take inspiration from the behavior of physical, biological, social, or artificial systems to perform optimization tasks without depending on the nature of the problem to solve [3]. Within these methods, a significant trend is swarm intelligence algorithms, in which the ability to find optimal values suddenly emerges by the simple interaction of agents in the search space [4].

Due to the No Free Lunch (NFL) Theorem [5], there is no metaheuristic that is able to optimize all kinds of problems better than the others. Thus, there is a need to continue devising new metaheuristics capable of optimizing issues of greater complexity and number of dimensions.

Cellular automata have been a highly recurrent model for understanding how complexity arises from a population

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

made up of simple elements interacting locally. The classical cellular automata model is discrete in time, space, and in the options of the possible states, each individual or cell may possess. However, despite their simplicity, which makes them easy to implement on a computer, they can produce complex emergent global behavior [6]. For this reason, cellular automata have been widely studied to analyze and simulate physical, chemical, and biological phenomena [7].

An interesting type of cellular automata is one whose overall behavior is reversible, provided a rule of evolution that indicates how the local interaction between the cells of the automaton takes place. This can be found as another rule to go back in the temporal dynamics of the system until the recovery of its original information. This kind of automata is known as reversible, and they have also been extensively studied to explain how local interactions can generate an invertible global behavior [8].

In this way, the diversity of behaviors published and investigated in cellular automata offers a huge source of inspiration to propose new metaheuristic optimization algorithms, analogous to how it has been done with the simplification and adoption of natural systems for the same purpose.

In this sense, reversible cellular automata also offer a new perspective to implementing a new metaheuristic algorithm given their dynamic properties. These can be summarized as their ability to preserve the initial information of the system, the possibility of generating all possible states, and that each state has a single ancestor, which can improve the exploration of a search space adequately.

This paper also proposes adapting the evolution rules in reversible elementary cellular automata (with only 2 states) to be applied to vectors of real values. First, the evolution rules are adapted to be applied with the information of two solutions or smart-cells to perform the exploration task. Then, another rule uses only the information contained in a single smart-cell to change it spontaneously and carry out the mission of exploiting solutions. This algorithm is named as reversible elementary cellular automata algorithm (RECAA).

The original part of this work continues to adapt various complex behaviors in cellular automata to propose metaheuristics for the global optimization of functions, as has been done previously in [9] and [10]. This adaptation is materialized in an algorithm based on two new rules for exploration and exploitation that are easy to implement.

RECAA is compared to other 14 metaheuristics widely recognized for their flexibility and performance in solving test optimization problems and engineering applications.

First, 50 test functions are taken for two experiments, one experiment with 30 dimensions and the other with 500 dimensions, using Wilcoxon statistical tests to compare and rank the algorithms [11]. Finally, RECAA is applied to 3 engineering problems of different dimensionality and used in the recent literature to compare their performance against well-known algorithms, obtaining satisfactory results.

The contributions of this work can be listed as follows:

- A new algorithm (RECAA) inspired by reversible cellular automata is presented for the global optimization of functions.
- RECAA is easy to implement and is available on the Github website at the link `https://github.com/juanseck/RECAA`.
- The efficient performance of the RECAA is demonstrated using test functions in multiple dimensions and comparing it with state-of-the-art metaheuristics.
- RECAA is used as well in engineering problems, getting satisfactory results.
- RECAA shows that cellular automata represent a vast source of inspiration for the proposal of new optimization metaheuristics.

This study is limited to optimize benchmark functions and engineering problems at the simulation level, no implementations for real cases are presented. Being a metaheuristic, RECAA will not have an efficient behavior in all problems as proved by the NFL theorem [5], and RECAA will not outperform an analytical method when it can be applied. It is important to note that RECAA is proposed for continuous problems; other types of combinatorial optimization problems, such as traveling salesman, routing, location, or scheduling problems are not addressed in this work and require an independent study.

On the other hand, the practical advantages lie in the easy implementation of the algorithm, which does not require adequate mathematical conditions to perform the optimization process, and continues with the research line of emerging cellular automata applications.

The manuscript is organized as follows. Section II shows a representative review of current and relevant works in swarm intelligence metaheuristics. Section III explains the operation and characteristics of reversible elementary cellular automata. Section IV describes the exploration and exploitation rules and the general operation of the proposed RECAA. Section V demonstrates the effectiveness of the RECAA using 50 test functions on 30 and 500 and with a fixed number of dimensions, comparing it statistically with other 14 state-of-the-art metaheuristics, obtaining satisfactory results. It also explains how the RECAA parameters were selected and experimentally shows their convergence from the exploration phase to the exploitation phase. Section VI applies RECAA to three engineering problems currently used to test and compare new metaheuristics, obtaining good results. Finally, Section VII presents the findings and suggests further work from this research.

II. REVIEW OF SWARM INTELLIGENCE METAHEURISTICS

The study of metaheuristics and intelligent algorithms has been expanded in many ways, with the creation of various algorithms inspired by different natural and social processes. These algorithms are based on evolutionary operations where a population of individuals is modified to a certain degree through iterations.

Metaheuristics and intelligent algorithms have been successfully used in many engineering applications due to their uncomplicated computational implementation and ability to solve complex problems. Intelligent algorithms can be divided into nine groups depending on their inspiration: biology-based, social-based, chemical-based, physics-based, music-based, mathematics-based, sports-based, swarm-based, plant-based, light-based, and water-based. The algorithm proposed in this work falls under the group of swarm-based or swarm intelligence algorithms (SI). References [12], [13] can be consulted for a detailed classification of intelligent algorithms.

Swarm intelligence algorithms (SI), which belong to a group of techniques based on the collective behavior of self-organized and decentralized systems. These systems are typically made up of a population of simple agents capable of perceiving and modifying their environment, making the communication possible between them, and simultaneously detecting changes in their peers' behavior. Although no centralized control structure usually dictates how agents should behave, local interactions between agents typically lead to the emergence of complex global behavior.

Some of the best-known algorithms include the particle swarm optimization (PSO) [14] which mimics the dynamics of birds using information exchange between individuals to find a better solution. Ant colony optimization (ACO) [15] which simulates the food gathering process that takes place in ant colonies and has been applied in many discrete problems. The artificial bee colony (ABC) [16] is based on the foraging behavior of honey bees.

As mentioned, SI mainly simulates the collective behavior of living organisms and uses social intelligence to obtain optimal solutions in the search space in a cooperative manner. All these algorithms share a basic structure of cooperative and competitive behaviors among their population members, using different model formulations, performance metrics, and adaptations to different problem scenarios.

Among the most recent SI algorithms, the Grasshopper Optimization Algorithm (GOA) stands out, inspired by the swarming behavior of grasshoppers. This algorithm has proven to be efficient in solving global unconstrained and constrained optimization problems. However, the original GOA has some drawbacks, such as the ease of falling into local optima and the slow convergence speed [17]. Another featured algorithm is the Whale Optimization Algorithm (WOA) for global search. The WOA is a metaheuristic algorithm that mimics the hunting behavior of humpback whales. It introduces the chaotic initialization phase for the whale swarm. Then, the Gaussian mutation is used to enhance the diversity in the population and uses a chaotic local search with a shrinking strategy to improve exploitation actions [18].

In [19], two new strategies were introduced in a synchronized way in the WOA, comprised of the Lévy flight previously used in the cuckoo search [20] and the chaotic local search to guide the swarm and promote the balance between the exploratory abilities and WOA search. In [21] Harris' hawk's optimization is explained, which imitates the cooperative behavior of birds. The HHO introduces two Gaussian mutation strategies to get away from local optima and a strategy of a dimensional decision previously applied in the cuckoo searching method to improve the speed of convergence. An improvement of HHO is proposed in [22] by integrating a logarithmic spiral-based exploration strategy with opposition learning and a local search technique with Rosebrock's method to obtain higher convergence accuracy. The Fruit Fly Optimization Algorithm (FOA) [23] offers the implementation of a simple and easy method. Nevertheless, for problems in many dimensions, the FOA result can be unsatisfactory and is prone to stagnation. An effective hunting strategy has been introduced in [24] to improve these aspects, inspired by whales to substitute the random searching plan of the original FOA. The monarch butterfly optimization (MBO) [25] is another relevant algorithm that simplifies the migration of monarch butterflies. Butterfly positions are updated by migration and adjustment operators. We also have the [26] slime mold algorithm (SMA), based on the oscillation mode of slime mold in nature. The proposed SMA has several new features and a unique mathematical model that uses adaptive weights to simulate the producing process of positive and negative feedback from the mold propagation wave to obtain an optimal path to get food.

In the hunger games search (HGS) [27], an optimization technique is proposed based on the behavior and activities driven by hungry animals. The HGS incorporates an adaptive weight to simulate the effect of starvation at each step of the search. In an attempt to go beyond methods based on physical or biological systems simplifications, there is a method based on the mathematical foundations of the Runge Kutta numerical algorithm known as the Runge Kutta (RUN) optimizer [28]. This method uses the logic of slope variations calculated by the RK method as a search mechanism for global optimization. Mathematical functions inspire another algorithm, the Sine-Cosine Algorithm (SCA) [29]. It has also been modified (MSCA) with control theory mechanisms, including the Cauchy mutation operator, the local chaotic search mechanism, in addition to opposition-based learning, and two operators supported on differential evolution [30]. Recently, inspired by war strategies, the War Strategy Optimization algorithm is proposed in [31], where the position of the solutions is updated using two war strategies and an adaptive mechanism that leads each soldier (or solution) to perform exploration and exploitation tasks in a balanced way.

All these metaheuristic algorithms share similar characteristics about searching stages that combine exploration and exploitation actions in a balanced way to obtain satisfactory results while solving different theoretical and physical problems. Similarly, recent works are inspired by the dynamic behavior of cellular automata, taking advantage of the great diversity of evolution rules which have been proposed and studied in recent decades. Examples of these optimization algorithms inspired by cellular automata for discrete problems include cellular particle swarm optimization

with a simple adaptive local search (CAPSO-SALS) [32], local neighborhood search algorithm (LNSA) [33], and the global-local neighborhood search algorithm (GLNSA) [34]. For continuous problems and searching global optima, there are also the cellular particle swarm optimization (CPSO) [35], [36], the continuous-state cellular automata algorithm (CCAA) [9] and the majority-minority cellular automata algorithm (MmCAA) [10].

This work seeks to contribute to this research field inspired by reversible behavior in elementary cellular automata in order to define a new global optimization metaheuristic that is easy to understand and implement, and shows competitive results against other recent metaheuristics already known for their efficient performance.

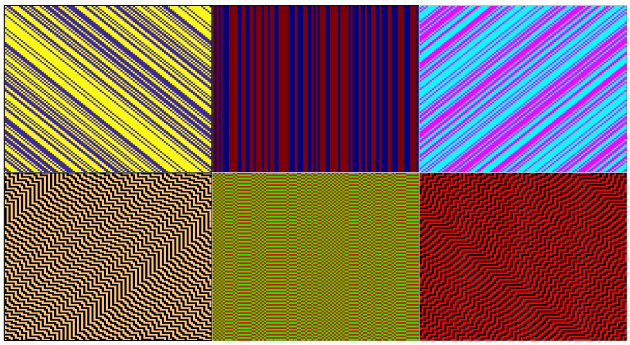

FIGURE 1. Evolutions of the six different RECAs, taking 100 cells, 100 time steps and different colors for 0 and 1 states.

III. BASICS ON REVERSIBLE CELLULAR AUTOMATA

A cellular automaton (CA) is an array of cells able to take a value from a finite set of states. All cells' array of states is a configuration of the CA. The simplest case is one-dimensional, where each cell checks its neighborhood (the current and its neighbor's state to each side) to establish its new state at the next step. The mapping of neighborhoods to states is known as the evolution rule. The evolution rule generates the CA dynamics by mapping one configuration onto another, establishing a discrete dynamic system both in its states, in the number of neighbors that each cell checks, and in the time steps that define the global behavior [6].

A one-dimensional CA is elementary (ECA) if each cell has only two possible states, and each cell checks only the state of its neighbor on each side to update its own state. It is customary to manage configurations as rings with periodic boundary conditions.

An ECA is reversible (RECA) if, given an evolution rule, another can be found to get back into the system dynamics. In other words, the evolution rule preserves the original information of the system, and this can be recovered by applying the inverse rule to get back to the initial configuration of the system.

There are 6 RECA rules, their local dynamics can be summarized as a cell considering only one neighbor's state, which can also be its own state, and copy or reverse it to update its state in the next time step (Fig. 1).

Thus, a RECA is characterized by each configuration having a single predecessor configuration, where each can be generated as part of the RECA evolution and not only as an initial configuration. In this way, a RECA can produce all possible configurations for a given number of cells [8].

Given the ability of a RECA to generate the entire configuration space, its behavior can serve as an inspiration to establish a new metaheuristic algorithm for global optimization.

IV. REVERSIBLE ELEMENTARY CELLULAR AUTOMATA ALGORITHM (RECAA)

RECAs can generate any global state in the configuration space by means of simple evolution rules. This property will be used to establish the behavior of the reversible elementary cellular automata algorithm (RECAA). Based on the evolution rules, we will define analogous rules that work on vectors of real numbers to establish exploration and exploitation actions, knowing that RECAs have the property of being able to generate all possible global states of the automaton.

The proposed RECAA first generates a random population of solutions or smart-cells, where at each iteration of the algorithm, each smart-cell generates a neighborhood with new positions.

Each smart-cell is a vector of real values representing a solution to the problem to be minimized. Each smart-cell will have as many elements as the number of unknowns or dimensions defining the problem to be optimized. Initially, these smart-cells are generated randomly, taking care that each real value belongs to the range of values allowed by the problem. Each smart-cell will change its values by applying rules inspired by the local behavior of RECAs.

In some cases, these rules will consider the information of another smart-cell to generate a new position (exploration), and in other cases, only the information of the same smart-cell will be taken to obtain a new solution (exploitation). In the end, the best value of the neighborhood of each smart-cell will be chosen, and if this solution improves the original smart-cell, its position will be updated.

As a population management strategy, elitism is used to conserve the best smart-cells in each iteration, and solutions worse than the original smart-cell will be accepted with a certain probability in order to maintain the diversity of the population and avoid stagnation of the solutions. The rules for performing the exploration and exploitation tasks of the proposed algorithm are presented below.

A. RULE FOR EXPLORATION USING TWO SMART-CELLS

The algorithm 1 presents the adaptation of the RECAs rule to obtain new neighbor solutions of a smart-cell $\boldsymbol{s}_i$ using another random smart-cell $\boldsymbol{s}_j$.

In the algorithm 1, first the resulting smart-cell ***evol*** is matched to the initial $\boldsymbol{s}_i$. Then the type of shift and whether to take the complementary values of $\boldsymbol{s}_j$ are selected

Algorithm 1 Exploration Rule

Result: New smart-cell ***evol***
Input: $\boldsymbol{s}_i, f(\boldsymbol{s}_i), \boldsymbol{s}_j, f(\boldsymbol{s}_j), prop, lb, ub$;
$\boldsymbol{evol} = \boldsymbol{s}_i$;
$shift\_type = randi(-1, 1)$;
$comp\_type = randi(0, 1)$;
$\boldsymbol{shift} = circshift(\boldsymbol{s}_j, shift\_type)$;
if $comp\_type = 1$ **then**
  $\boldsymbol{shift} = up - \boldsymbol{shift} + lb$;
end
$sum = f(\boldsymbol{s}_i) + f(\boldsymbol{s}_j)$;
$pond = 1 - (f(\boldsymbol{s}_j)/sum)$;
$r = (rand \cdot prop) - (prop/2)$;
forall the k *in length*(***evol***) **do**
  if $rand <= pond$ **then**
    $\boldsymbol{evol}(k) = \boldsymbol{evol}(k) + (r \cdot \boldsymbol{shift}(k))$;
  end
end

randomly. The *randi*() function randomly selects an integer value between the first and second parameters.

The shift of the neighboring smart-cell is indicated by the *circshift*() function, which performs a circular shift of $\boldsymbol{s}_j$ from one position to the left (type -1), no shift (type 0) or one position to the right (type 1), analogously as done in the evolution rules of RECAs.

The complementary value means taking the distance $\boldsymbol{shift} - lb$ from the upper bound ub and is another adaptation of the action of the RECAs rules that exchange binary values but now operate with real values. The rule will have a higher probability of changing the values of *bmevol* the larger $f(\boldsymbol{s}_i)$ is with respect to $f(\boldsymbol{s}_j)$.

To finish, in the last cycle the chosen values of *bmevol* are transformed by adding a random proportion r of the values at the same positions in *bmshift*, where r ranges from $-(prop/2)$ to $(prop/2)$. This procedure is the proposed adaptation of the RECAs rules to handle real values and will be employed in the RECAA to perform the exploration of new solutions in the search space. Figure 2 illustrates the operation of the exploration rule.

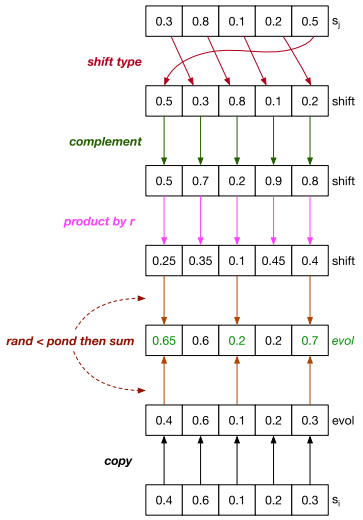

FIGURE 2. Exploration rule for a smart-cell with 5 values.

Algorithm 2 Exploitation Rule

Result: New smart-cell ***evol***
Input: $\boldsymbol{s}_i, prop, lb, ub$;
$\boldsymbol{evol} = \boldsymbol{s}_i$;
$shift\_type = randi(-1, 1)$;
if $shift\_type = 0$ **then**
  $comp\_type = 1$;
else
  $comp\_type = randi(0, 1)$;
end
$\boldsymbol{shift} = circshift(\boldsymbol{s}_i, shift\_type)$;
if $comp\_type = 1$ **then**
  $\boldsymbol{shift} = up - \boldsymbol{shift} + lb$;
end
$r = (rand \cdot prop)$;
$\boldsymbol{diff} = (\boldsymbol{s}_i - \boldsymbol{shift}) \cdot r$;
$\boldsymbol{evol} = \boldsymbol{evol} - \boldsymbol{diff}$;

B. RULE FOR EXPLOITATION USING THE INFORMATION OF A SMART-CELL

The algorithm 2 shows the adaptation of the reversible rules to obtain a new neighbor solution of a smart-cell $\boldsymbol{s}_i$ using only its information.

In the algorithm 2, first match the resulting smart-cell ***evol*** with the initial $\boldsymbol{s}_i$. The type of shift and the option to take the complementary values of $\boldsymbol{s}_i$ are specified at random.

The smart-cell shifts and the complementary value is performed if necessary, similar to the RECAs evolution rules. A random value r between 0 and *prop* is taken. A vector ***diff*** of differences between the values of the smart-cell and the obtained shift is calculated and weighted by r. The values *bmevol* are finally acquired by subtracting them from the values obtained in *bmdiff*. This process is the proposed adaptation for the information exploitation of each smart-cell. Figure 3 exemplifies the operation of the exploitation rule.

C. RULE FOR ROUNDING VALUES IN A SMART-CELL

The rule in algorithm 3 is used to round randomly selected values of a smart-cell to a fixed number of decimal values n_r. Depending on the actual value $f(\boldsymbol{s}_i)$, if it is close to the best obtained cost $f(\boldsymbol{b}_S)$, then there will be a higher chance of rounding. This rule was introduced in [9] and also used in [10] and is very useful for bounding solutions in the later stages of the optimization process, especially for engineering design problems. Figure 4 depicts the operation of the rounding rule.

D. COMPLETE STRUCTURE OF THE RECAA

The complete pseudocode of RECAA is presented in Algorithm 4. RECAA receives as input the number of smart-cells n_s, the number of neighbors per smart-cell n_{ne}, the number of iterations n_{it}, the number of elitist solutions n_{el}, the lower bound lb and upper bound ub for the elements of each smart-cell, the number of dimensions n_d and the functions f

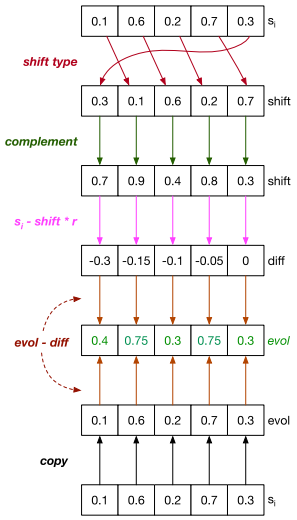

FIGURE 3. Exploitation rule for a smart-cell with 5 values.

Algorithm 3 Rounding Rule

Result: New smart-cell ***evol***
Input: $\boldsymbol{s}_i, f(\boldsymbol{s}_i), f(\boldsymbol{b}_S), n_r$;
$\boldsymbol{evol} = \boldsymbol{s}_i$;
$sum = f(\boldsymbol{s}_i) + f(\boldsymbol{b}_S)$;
$pond = 1 - (f(\boldsymbol{s}_i)/sum)$;
forall the k *in length*(***evol***) **do**
  if $rand <= pond$ **then**
    $\boldsymbol{evol}(k) = round(\boldsymbol{evol}(k), n_r)$;
  end
end

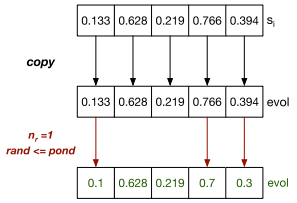

FIGURE 4. Rounding rule for a smart-cell with 5 values.

to optimize. First, the parameters for the operation of RECAA are defined. The proportion *prop* in which the effect of the evolution rules will be applied, the parameters $lower_r$ and $upper_r$ for the rounding rule, the probabilities $prob_\alpha$ and $prob_\beta$ to select the type of rule to apply for each smart-cell in each iteration and $prob_\gamma$ to be able to choose solutions that do not improve the current solution in order to avoid smart-cell stagnation.

The initial population S is randomly generated, and each solution is evaluated at f. In the optimization cycle, first, the best n_{el} smart-cells are kept, then for the rest, a neighborhood of n_{ne} neighbors is generated by randomly selecting a rule according to the probabilities $prob_\alpha$ and $prob_\beta$ and the best solution is taken from each neighborhood. The new solution replaces the original smart-cell if it improves its fitness value in f or with a probability $prob_\gamma$ otherwise. In the end, the best smart-cell and its fitness value are returned.

To comply with the restrictions on upper and lower bounds allowed for the optimal solutions, when RECAA obtains smart-cells with a value that exceeds the allowed upper limit ub, the algorithm returns the value to the valid interval by selecting a random number between $[ub - (ub - lb)/4, ub]$. The analogous process is done in case a value falls below lb by taking a random value between $[lb, lb + (ub - lb)/4]$. This process has been previously implemented in similar algorithms [9], [10], [36], obtaining satisfactory results. Figure 5 shows the flow chart of the proposed algorithm.

Algorithm 4 Reversible Elemental Cellular Automata Algorithm (RECAA)

Result: Best smart-cell $\boldsymbol{b}_S$ and fitness value $f(\boldsymbol{b}_S)$
Input: $n_S, n_{ne}, n_{it}, n_{el}, lb, ub, n_d, f$;
Set parameters *prop*, $lower_r$, $upper_r$, $prob_\alpha$, $prob_\beta$, $prob_\gamma$;
Generate random population S of n_S smart-cells;
Evaluate S in f;
forall the $i = 2$ *to* n_{it} **do**
  Keep the best n_{el} smart-cells in a new population;
  forall the $j = n_{el} + 1$ *to* n_S **do**
    Take $\boldsymbol{s}_j$ and another random smart-cell $\boldsymbol{s}_r$ from S for the rule requiring an extra solution;
    forall the $k = 1$ *to* n_{ne} **do**
      Choose a random rule R from Algorithm 1 with probability $prob_\alpha$, Algorithm 2 with probability $prob_\beta$ and Algorithm 3 with the remaining probability ;
      Obtain $\boldsymbol{evol}_k =$ $R(\boldsymbol{s}_j$, additional parameters of the rule);
      Check that the n_d values of $\boldsymbol{evol}_k$ are between lb and ub and correct if necessary;
      Calculate $f(\boldsymbol{evol}_k)$;
    end
    Choose the best neighbor ***evol*** from the k generated neighbors having a minimum cost $f(\boldsymbol{evol})$;
    if $rand < prob_\gamma$ *or* $f(\boldsymbol{s}_j) > f(\boldsymbol{evol})$ **then**
      $\boldsymbol{s}_j = \boldsymbol{evol}$;
    end
  end
end
Return the best smart-cell $\boldsymbol{b}_S$ and its fitness value $f(\boldsymbol{b}_S)$;

E. COMPLEXITY ANALYSIS OF THE RECAA

The logical operations of RECAA are uncomplicated, so its computational complexity is low. The following analysis is calculated with big-$\mathcal{O}$ notation. The initialization of the n_S smart-cells has a complexity of $\mathcal{O}(n_S \cdot n_d)$. The evaluation of each function is $\mathcal{O}(n_S)$, the elitist handling of solutions

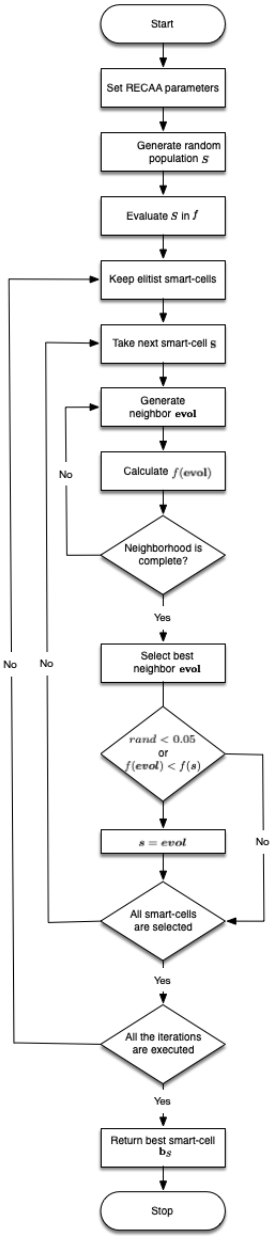

FIGURE 5. Flow chart of the RECAA.

implies saving the positions of the best n_{el} smart-cells. This process can be done simultaneously as the function evaluation, preserving the complexity of $\mathcal{O}(n_S)$. Updating the values of each smart-cell depends on taking n_{ne} neighbors and applying each randomly selected rule. The three rules have complexity $\mathcal{O}(n_d)$, so the update of the smart-cells is given by $\mathcal{O}(n_s \cdot n_{ne} \cdot n_d)$. Thus, the total complexity of RECAA is $\mathcal{O}(n_{it} \cdot n_s \cdot n_{ne} \cdot n_d)$. This complexity is similar to the computational cost of many newer algorithms proposed for global optimization [12], [13], [31].

V. EXPERIMENTATION WITH BENCHMARK FUNCTIONS IN DIFFERENT DIMENSIONS

Table 1 presents the 50 test functions used to test the effectiveness of RECAA. The first 16 functions are unimodal

TABLE 1. Benchmark functions.

No.	Name	No.	Name
F1	Brown	F26	Penalty 2
F2	Chung-Reynolds	F27	Rastrigin
F3	Exponential	F28	Salomon
F4	Powell 1	F29	Schwefel 2.26
F5	Powell 2	F30	Shubert 3
F6	Powell Sum	F31	Streched V Sine Wave
F7	Quartic	F32	Wavy 1
F8	Ridge	F33	Xin-She Yang
F9	Schwefel 1.2	F34	Beale
F10	Schwefel 2.2	F35	Corana
F11	Schwefel 2.2	F36	Cross-in-Tray
F12	Schwefel 2.2	F37	Drop-Wave
F13	Schwefel 2.2	F38	Foxholes
F14	Sphere	F39	Hartman 1
F15	Step 2	F40	Hartman 2
F16	Sum Squares	F41	Helical Valley
F17	Ackley 1	F42	Hump
F18	Alpine 1	F43	Kowalik
F19	Alpine 2	F44	Matyas
F20	Conditioned Elliptic	F45	Miele-Cantrell
F21	Cosine Mixture	F46	Shekel 5
F22	Griewank	F47	Shekel 7
F23	Levy	F48	Shekel 10
F24	Pathological	F49	Watson
F25	Penalty 1	F50	Wolfe

and are used to test the exploitability of metaheuristics. The following 17 multimodal functions test metaheuristics exploration properties and local minima escapability. The final 17 functions have a fixed dimension and contemplate several local minima to test the trade-off between an algorithm's exploration and exploitation actions. The specification and properties of these functions can be found in [11] and [37], and their computational implementation can be reviewed from http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/go.htm and http://www.sfu.ca/~ssurjano/optimization.html.

The algorithms that were taken to perform the computational experiments are the aquila optimizer (AO) [38], the Archimedes optimization algorithm (AOA) [39], the heap-based optimizer (HBO) [40], the hunger game search (HGS) [27], the Harris hawk optimization (HHO) [21], the adaptive differential evolution based on the success history adaptive differential evolution with linear reduction in population size (LSHADE) [41], the LSHADE with semiparameter adaptation hybrid with CMA-ES (LSPACMA) [42], the marine predator algorithm (MPA) [43], the modified sine cosine algorithm (MSCA) [30], the political optimizer (PO) [44], the pure randomized orthogonal search (PROS) algorithm [45], the Pareto-like sequential sampling (PSS) heuristic [46], the slime mould algorithm (SMA) [26], and the weighted superposition attraction (WSA) [47].

These algorithms have been published recently and are recognized for their effectiveness in optimizing unimodal and multimodal problems in multiple dimensions. Thus, they represent a suitable set of metaheuristics for analyzing and

TABLE 2. Parameter settings of algorithms employed for comparison with RECAA.

Algorithm	Parameters
AO	$\alpha = 0.1, \delta = 0.1$
AOA	$C1 = 2, C2 = 6, C3 = 2, C4 = 0.5$
HBO	$C = \lfloor n_{it}/25 \rfloor, p_1 = 1 - (t/n_{it})$
HGS	$l = 0.08, LH = 10000$
HHO	$E_0 = 2 \cdot rand - 1, J = 2 \cdot (1 - rand)$
LSHADE	$rate = 0.11, arc = 1.4, mem = 5$
LSPACMA	$L = 0.8, rate = 0.11, arc = 1.4, mem = 5$
MPA	$P = 0.5, FADs = 0.2$
MSCA	$a = 2, b = 0.5, r = a(sin((1 - (t/n_{it})) \cdot (\pi/2))) + b$
PO	$n = 8, \lambda : 1 \rightarrow 0$
PROS	Parameterless algorithm
PSS	$\alpha = 0.95$
SMA	$z = 0.03$
WSA	$\tau = 0.8, sl_o = 0.035, \lambda = 0.75, \varphi = 0.001$

weighting RECAA against state-of-the-art algorithms. The parameter settings for these fourteen algorithms are taken from their original papers and are given in Table 2.

The original Matlab implementations of these algorithms were taken directly from the web addresses indicated in the reference articles. The Matlab code of RECAA can be downloaded on Github from the link https://github.com/juanseck/RECAA.

RECAA and the other algorithms were executed in Matlab 2015a on a 3.1 GHz Intel Xeon CPU and 64 GB in RAM with a macOS Big Sur operating system. 30 independent runs were made for each algorithm on every benchmark function.

RECCA was executed with $n_S = 12$ and $n_{ne} = 6$, for the rest of the algorithms $(n_S) = 72$ individuals were used, all algorithms applied $n_{it} = 500$ iterations. The starting search points in the optimization process are defined by every algorithm, respecting the original implementation and favoring a fair comparison.

A. PARAMETER TUNNING OF RECAA

In the experiments performed, the number of smart-cells, neighbors, and iterations were fixed with the values mentioned above. For the rest of the parameters that define the behavior of the rules in RECAA (elitism n_{el}, ratio *prop* and rounding limits $lower_r$ and $upper_r$), an experiment was performed with three levels for each parameter, taking the test functions F_{14} (unimodal), F_{30} (multimodal) in 30 dimensions, and F_{43} with fixed dimensions to select the combination of parameters with the best minimization results.

For the elitism part, 1, 2 and 3 levels were tested for n_{el}; for the *prop* ratio used in the RECAA rules, three values 1.4, 1.7 and 2 were experimented. To establish the rounding value n_r in the 3 rule, the values 0, 1, and 2 for $lower_r$ as the lower bound and the values 5, 6, and 7 for $upper_r$ defining the upper rounding bound were tried. In total, 81 combinations were tried, each with 30 independent runs on the test functions to obtain the most suitable set of parameters utilizing the best average value. The parameters selected for comparative testing with other state-of-the-art algorithms are presented in Table 3.

TABLE 3. Parameter settings of the RECAA.

Number of smart-cells (n_S)	12
Number of neighbors (n_{ne})	6
Number of iterations (n_{it})	500
Number of elitist solutions (n_{el})	2
Rule parameters	
Proportion ($prop$)	1.7
Lower rounding ($lower_r$)	2
Upper rounding ($upper_r$)	6

B. CONTRIBUTION OF RULES IN THE EXPLORATION AND EXPLOITATION OF THE SEARCH SPACE

Three different types of rules are used in RECAA, one from Algorithm 1 for exploration of the search space by exchanging information between smart-cells and those from Algorithms 2 and 3 for exploitation of the information of each smart-cell individually.

The following experiment was performed to visualize the type of rules acting more relevant during optimization. For some selected test functions, RECAA was run, and at each iteration, the best neighbor was observed, and the rule used at the time was kept when it improves the smart-cell. If the rule is an exploration rule, a value of 10 was accumulated. If it is an exploitation rule, only a value of 1 was accumulated. For a proper optimization process, a high value should be observed in the record of rules that improve a solution indicating an intense exploration phase and then decrease, indicating the change to the exploitation phase of solutions.

This experiment was applied to the same functions employed to tune the RECAA parameters (F_{14}, F_{30}, F_{43}) making 30 independent runs and taking the average of the rule values that caused smart-cell improvements in each iteration. The results are presented in Fig. 6.

In the experiment, it can be observed that the action of the exploration rule is more intense in the initial iterations of the algorithm, to then move on to the exploitation stage where the rules of the algorithms 2 and 3 become more significant to finish the optimization process.

C. FIRST EXPERIMENT WITH BENCHMARK FUNCTIONS IN 30 AND FIXED DIMENSIONS

In this experiment, the first 33 test functions were taken in 30 dimensions, and the following 17 functions are fixed-dimensional according to Table 1. 30 independent runs were made for each algorithm, saving the average value and standard deviation for each case. Table 5 shows the results obtained for the unimodal, multimodal, and fixed-dimensional functions, where the best results for all algorithms are presented in bold with an orange background.

RECAA calculated 15 of the 16 best average values for the unimodal functions, performing similar to MSCA and outperforming the rest of methods. RECAA also obtained 15 of the best values for standard deviation, demonstrating its exploitability. RECAA achieved 11 of the 17 best

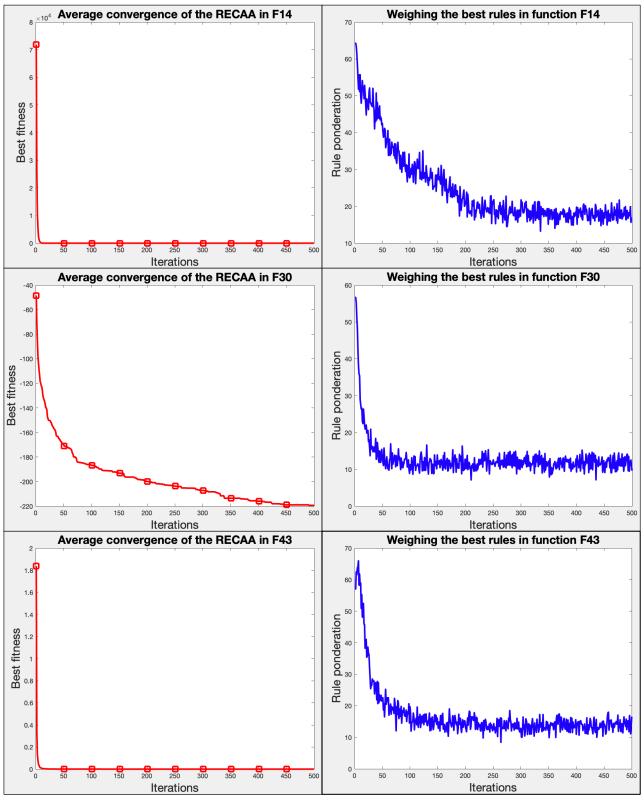

FIGURE 6. Rules weighted in the optimization process to show the transition from exploration to exploitation.

average values for the multimodal case. This result is only surpassed by the PO, which obtained 13 best average values; the other methods were below RECAA. RECAA also obtained 11 better values in the standard deviation, proving its capacity to exploit information. In the case of fixed-dimensional functions, RECAA computed 6 best average values, being outperformed by the HBO, HGS, LSHADE, and LSPACMA algorithms and with a similar number of best results as MPA and PO. RECAA generated 6 better values concerning standard deviation, showing a competitive balance between information exploration and exploitation for fixed-dimensional problems.

Table 4 presents the results of the Wilcoxon rank-sum statistical test comparing RECAA with each of the other methods for each test function, where the symbol + represents a better result that is statistically significant, the symbol *approx* indicates no significant difference, and − indicates a worse statistically significant result. The Avg column lists each algorithm's average rank obtained by optimizing the test functions, and the Rank column shows the order in which each algorithm is ranked according to its average. RECAA obtained the second-best rank behind PO. It should be noted that RECAA outperformed PO by 14 test functions and was outperformed by fewer functions (13) by this algorithm. For the rest of the methods, RECAA obtained a more significant difference with respect to the number of functions with a better result in this experiment.

TABLE 4. Wilcoxon rank-sum test and ranking of the compared algorithms on 30D and fixed problems.

Algorith	+/−/≈	Avg	Rank
AO	33/3/14	5.16	8
AOA	31/1/18	6.92	12
HBO	32/7/11	6.06	10
HGS	22/8/20	3.41	3
HHO	28/4/18	4.92	7
LSHADE	34/11/5	6.31	11
LSPACMA	33/10/7	6.04	9
MPA	28/14/8	4.54	6
MSCA	16/4/30	3.44	4
PO	14/13/23	2.52	1
PROS	47/2/1	9.58	14
PSS	47/1/2	10.11	15
RECAA	–/–/–	2.86	2
SMA	19/10/21	3.52	5
WSA	47/1/2	8.78	13

D. SECOND EXPERIMENT WITH BENCHMARK FUNCTIONS IN 500 AND FIXED DIMENSIONS

In this case, the 33 unimodal and multimodal functions were taken in 500 dimensions, and the last 17 functions were in fixed dimensional. For each algorithm, another 30 independent runs were run, calculating the average value and standard deviation. The best results for all algorithms are shown in bold with an orange background (Table 6).

RECAA obtained 13 of the best average values for the unimodal functions, performing similar to MSCA and outperforming the other algorithms. RECAA obtained 14 minimum values for standard deviation, proving its exploitability for higher dimensions.

RECAA achieved 12 out of 17 best average values for the multimodal case. In this experiment, the other algorithms fell below RECAA. RECAA generated 12 minimum values in the standard deviation, confirming its good information exploitation. RECAA obtained 5 of the best average values for the fixed-dimensional functions, with a performance below the HBO, HGS, LSHADE, LSPACMA, MPA, and PO algorithms. RECAA generated 5 better values concerning the standard deviation, again showing an acceptable balance between exploration and exploitation actions in this type of problem.

Table 7 shows the Wilcoxon rank-sum statistical test comparing RECAA with the other methods. RECAA again obtained the second-best rank behind PO. In general, RECAA obtained a more significant or equal difference (in the case of PO) with respect to the number of functions with a better result in this experiment. Fig. 7 shows some examples of the convergence curves for different test functions in 30 and 500 dimensions, as well as in the functions with fixed dimensions. Only the algorithms with the best results are presented in these examples.

E. THIRD EXPERIMENT WITH CEC 2022 BENCHMARK FUNCTIONS

In the last experiment, the CEC 2022 set of 12 benchmark functions were taken in 20 dimensions, including shifted,

TABLE 5. Metaheuristics compared with RECAA on 30 and fixed dimensional problems.

Fn	Stats	AO	AOA	HBO	HGS	HHO	LSHADE	LSPACMA	MPA	MSCA	PO	PROS	PSS	RECAA	SMA	WSA
F1	AVG	6.9e-103	2.3e+00	1.9e-12	5.0e-235	1.3e-105	3.4e-15	1.5e-29	1.2e-25	**0.0e+00**	**0.0e+00**	4.7e-04	3.69e-03	**0.0e+00**	**0.0e+00**	6.3e-14
	STD	3.5e-102	1.2e+00	1.4e-12	**0.0e+00**	9.5e-105	8.2e-15	2.0e-29	1.2e-25	**0.0e+00**	**0.0e+00**	1.9e-04	1.5e-02	**0.0e+00**	**0.0e+00**	7.3e-15
F2	AVG	6.9e-302	4.3e-11	6.9e-19	**0.0e+00**	5.5e-203	5.2e-24	6.7e-36	4.5e-45	**0.0e+00**	**0.0e+00**	1.9e-01	2.9e+02	**0.0e+00**	**0.0e+00**	8.1e-22
	STD	**0.0e+00**	3.1e-10	1.1e-18	**0.0e+00**	**0.0e+00**	2.1e-23	3.0e-35	8.9e-45	**0.0e+00**	**0.0e+00**	2.0e-01	1.6e+03	**0.0e+00**	**0.0e+00**	1.4e-22
F3	AVG	**-1.0e+00**	**-1.0e+00**	-1.0e+00	**-1.0e+00**	**-1.0e+00**	-1.0e+00	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	-1.0e+00	-1.0e+00	**-1.0e+00**	**-1.0e+00**	-1.0e+00
	STD	**0.0e+00**	**0.0e+00**	2.8e-14	**0.0e+00**	**0.0e+00**	1.5e-16	1.5e-17	**0.0e+00**	**0.0e+00**	**0.0e+00**	8.9e-06	1.8e-04	**0.0e+00**	**0.0e+00**	2.6e-16
F4	AVG	1.4e-104	1.2e-01	3.3e-02	2.9e-37	1.3e-100	3.5e-05	2.1e-06	1.9e-14	**0.0e+00**	**0.0e+00**	1.2e-01	7.4e-01	1.2e-04	**0.0e+00**	8.5e-13
	STD	8.1e-104	5.5e-01	1.8e-02	2.0e-36	9.4e-100	3.1e-05	1.8e-06	1.3e-13	**0.0e+00**	**0.0e+00**	4.9e-02	7.7e-01	3.3e-04	**0.0e+00**	1.7e-13
F5	AVG	1.2e-105	9.7e-05	4.7e-06	2.2e-237	2.7e-104	3.3e-09	3.4e-15	5.7e-20	**0.0e+00**	**0.0e+00**	8.5e-02	1.1e+00	**0.0e+00**	**0.0e+00**	5.8e-12
	STD	6.9e-105	6.8e-04	3.9e-06	**0.0e+00**	1.8e-103	6.1e-09	1.0e-14	3.3e-19	**0.0e+00**	**0.0e+00**	3.1e-02	3.7e+00	**0.0e+00**	**0.0e+00**	5.8e-13
F6	AVG	1.6e-22	**0.0e+00**	2.5e-31	1.1e-104	8.9e-126	2.5e-36	5.7e-30	1.0e-62	**0.0e+00**	**0.0e+00**	1.3e-06	4.5e-08	**0.0e+00**	**0.0e+00**	4.4e-19
	STD	1.1e-21	**0.0e+00**	1.0e-30	7.5e-104	6.3e-125	1.2e-35	3.6e-29	4.9e-62	**0.0e+00**	**0.0e+00**	2.2e-06	3.1e-08	**0.0e+00**	**0.0e+00**	4.8e-19
F7	AVG	1.7e-300	**0.0e+00**	3.3e-18	**0.0e+00**	4.2e-204	1.2e-25	1.6e-55	2.9e-48	**0.0e+00**	**0.0e+00**	1.0e-08	2.0e-06	**0.0e+00**	**0.0e+00**	2.2e-29
	STD	**0.0e+00**	**0.0e+00**	7.1e-18	**0.0e+00**	**0.0e+00**	3.1e-25	5.7e-55	5.6e-48	**0.0e+00**	**0.0e+00**	1.0e-08	7.2e-06	**0.0e+00**	**0.0e+00**	3.9e-30
F8	AVG	-3.4e+00	**-5.0e+00**	-5.0e+00	-5.0e+00	-5.0e+00	-5.0e+00	-5.0e+00	-5.0e+00	-2.6e+00	-4.4e+00	-4.9e+00	-4.9e+00	**-5.0e+00**	-5.0e+00	1.9e-07
	STD	2.9e-01	**0.0e+00**	4.1e-07	**0.0e+00**	3.6e-10	5.9e-07	8.6e-14	5.4e-07	2.5e+00	2.7e-01	1.2e-02	4.2e-02	**0.0e+00**	2.7e-06	1.7e-08
F9	AVG	6.3e-109	4.1e-04	2.0e+04	3.1e-140	6.4e-85	2.2e-01	1.1e-02	3.4e-05	**0.0e+00**	**0.0e+00**	6.7e+03	5.2e+01	**0.0e+00**	2.3e-315	5.4e-11
	STD	4.4e-108	1.7e-03	5.7e+03	1.9e-139	4.5e-84	2.4e-01	1.5e-02	5.7e-05	**0.0e+00**	**0.0e+00**	1.8e+03	1.9e+01	**0.0e+00**	**0.0e+00**	1.3e-11
F10	AVG	1.0e-57	1.9e-74	1.5e-06	1.2e-123	7.4e-53	9.4e-06	2.9e-06	2.7e-12	**0.0e+00**	6.2e-202	2.5e+00	8.6e+00	**0.0e+00**	4.8e-158	2.4e-05
	STD	7.2e-57	1.2e-73	6.0e-07	8.5e-123	4.9e-52	1.0e-05	6.0e-06	2.1e-12	**0.0e+00**	**0.0e+00**	4.4e-01	1.2e+01	**0.0e+00**	2.3e-157	1.4e-06
F11	AVG	3.3e-77	1.6e-02	3.69e+00	6.7e-117	2.2e-52	1.2e-01	6.5e-02	2.7e-09	**0.0e+00**	1.3e-175	2.6e+00	1.7e+00	**0.0e+00**	4.7e-168	1.8e-06
	STD	2.3e-76	1.9e-02	1.3e+00	4.7e-116	1.0e-51	8.7e-02	7.4e-02	1.5e-09	**0.0e+00**	**0.0e+00**	5.4e-01	1.6e+00	**0.0e+00**	**0.0e+00**	7.0e-08
F12	AVG	1.7e-60	9.2e-65	1.5e-06	9.0e-125	5.8e-51	4.2e-02	4.8e-02	3.3e-12	**0.0e+00**	2.4e-201	2.6e+00	1.5e+24	**0.0e+00**	4.3e-153	2.4e-05
	STD	1.2e-59	6.5e-64	5.6e-07	4.9e-124	4.1e-50	2.8e-01	1.0e-01	2.9e-12	**0.0e+00**	**0.0e+00**	5.0e-01	1.0e+25	**0.0e+00**	3.0e-152	1.5e-06
F13	AVG	**0.0e+00**	1.0e-15	4.9e-12	**0.0e+00**	**0.0e+00**	3.2e-25	6.3e-34	2.1e-92	**0.0e+00**	**0.0e+00**	1.6e-03	3.1e+06	**0.0e+00**	**0.0e+00**	1.3e-57
	STD	**0.0e+00**	4.4e-15	2.2e-11	**0.0e+00**	**0.0e+00**	1.7e-24	4.4e-33	1.3e-91	**0.0e+00**	**0.0e+00**	8.6e-03	2.2e+07	**0.0e+00**	**0.0e+00**	4.6e-58
F14	AVG	2.0e-138	2.4e-57	9.2e-10	1.1e-242	9.0e-103	5.6e-13	1.0e-18	5.0e-23	**0.0e+00**	**0.0e+00**	4.0e-01	2.6e+00	**0.0e+00**	2.5e-313	2.8e-11
	STD	1.4e-137	1.7e-56	7.9e-10	**0.0e+00**	5.3e-102	9.0e-13	3.1e-18	5.6e-23	**0.0e+00**	**0.0e+00**	1.4e-01	1.0e+01	**0.0e+00**	**0.0e+00**	2.5e-12
F15	AVG	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	6.2e-01	6.6e-01	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.4e-01	1.0e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**
	STD	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	7.8e-01	9.3e-01	**0.0e+00**	**0.0e+00**	**0.0e+00**	3.5e-01	1.1e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**
F16	AVG	3.1e-144	4.8e-38	1.2e-08	5.5e-247	6.4e-99	1.7e-11	1.6e-16	6.8e-22	**0.0e+00**	**0.0e+00**	6.5e+00	4.1e+01	**0.0e+00**	1.8e-287	4.0e-10
	STD	2.2e-143	3.4e-37	1.2e-08	**0.0e+00**	4.5e-98	3.0e-11	6.5e-16	5.5e-22	**0.0e+00**	**0.0e+00**	2.5e+00	9.69e+01	**0.0e+00**	**0.0e+00**	5.0e-11
F17	AVG	**8.8e-16**	**8.8e-16**	7.9e-06	**8.8e-16**	**8.8e-16**	3.4e-01	5.4e-12	1.7e-12	**8.8e-16**	**8.8e-16**	2.4e-01	1.9e+00	**8.8e-16**	**8.8e-16**	1.3e-06
	STD	**0.0e+00**	**0.0e+00**	3.7e-06	**0.0e+00**	**0.0e+00**	5.2e-01	9.3e-12	9.3e-13	**0.0e+00**	**0.0e+00**	6.6e-02	1.2e+00	**0.0e+00**	**0.0e+00**	6.9e-08
F18	AVG	9.9e-06	**0.0e+00**	8.5e-05	8.4e-127	7.9e-37	4.1e-06	1.0e-04	8.1e-14	2.9e-09	1.7e-203	1.1e-02	5.2e+00	**0.0e+00**	1.2e-164	2.4e-07
	STD	3.7e-05	**0.0e+00**	2.2e-04	6.0e-126	5.5e-36	1.0e-05	4.9e-04	7.4e-14	2.0e-08	**0.0e+00**	2.2e-03	5.3e+00	**0.0e+00**	**0.0e+00**	1.1e-08
F19	AVG	-2.8e+13	-7.9e+08	-2.7e+13	-2.8e+13	-1.7e+13	-4.9e+12	-2.0e+13	-3.1e+12	-2.8e+13	**-2.8e+13**	-2.1e+12	-1.9e+13	-2.8e+13	-2.8e+13	-3.2e+05
	STD	6.4e+10	7.6e+08	2.69e+12	8.4e+09	1.2e+13	2.1e+12	7.5e+12	5.3e+12	8.3e+08	**3.9e-03**	1.7e+12	5.6e+12	3.3e+09	4.5e+08	4.7e+05
F20	AVG	1.1e-106	2.5e-72	2.2e-06	1.1e-281	3.9e-97	5.1e-07	1.4e-03	2.2e-19	**0.0e+00**	**0.0e+00**	4.0e+04	7.7e+05	**0.0e+00**	1.4e-158	8.9e-07
	STD	7.4e-106	1.8e-71	1.8e-06	**0.0e+00**	2.7e-96	1.9e-06	5.7e-03	2.3e-19	**0.0e+00**	**0.0e+00**	3.9e+04	1.7e+06	**0.0e+00**	9.9e-158	2.3e-07
F21	AVG	**0.0e+00**	**0.0e+00**	7.4e-13	**0.0e+00**	**0.0e+00**	1.1e-02	8.8e-03	**0.0e+00**	**0.0e+00**	**0.0e+00**	5.6e-04	1.5e-03	**0.0e+00**	**0.0e+00**	3.8e-14
	STD	**0.0e+00**	**0.0e+00**	5.9e-13	**0.0e+00**	**0.0e+00**	4.0e-02	3.5e-02	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.3e-04	5.6e-03	**0.0e+00**	**0.0e+00**	4.2e-15
F22	AVG	**0.0e+00**	**0.0e+00**	9.5e-10	**0.0e+00**	**0.0e+00**	4.6e-03	2.1e-03	**0.0e+00**	**0.0e+00**	**0.0e+00**	3.6e-02	2.7e-01	**0.0e+00**	**0.0e+00**	1.4e-12
	STD	**0.0e+00**	**0.0e+00**	4.3e-09	**0.0e+00**	**0.0e+00**	6.8e-03	4.0e-03	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.3e-02	3.8e-01	**0.0e+00**	**0.0e+00**	1.7e-13
F23	AVG	5.7e-05	2.4e+00	1.5e-11	7.1e-07	9.3e-06	1.5e-01	1.0e-27	5.9e-02	1.1e-07	**1.5e-32**	2.0e-03	2.3e-02	1.5e-02	4.0e-04	2.8e+00
	STD	1.1e-04	9.1e-02	1.2e-11	1.1e-06	1.3e-05	2.0e-01	1.4e-27	6.1e-02	1.6e-07	**0.0e+00**	7.4e-04	1.0e-01	8.3e-02	3.6e-04	2.6e-01
F24	AVG	2.6e-07	**0.0e+00**	3.1e+00	2.2e-05	1.4e-05	7.4e+00	7.2e+00	1.4e+00	3.1e-07	7.69e-04	2.9e+00	1.1e+01	**0.0e+00**	1.1e-04	3.8e-07
	STD	5.6e-07	**0.0e+00**	4.6e-01	5.9e-05	3.6e-05	3.5e-01	4.2e-01	9.8e-01	5.3e-07	2.3e-03	4.3e-01	4.1e-01	**0.0e+00**	1.7e-04	2.9e-07
F25	AVG	4.8e-07	3.9e-01	4.1e-11	3.2e-09	1.1e-06	8.2e-03	6.2e-03	7.0e-10	1.0e-08	**1.5e-32**	2.5e-03	2.2e+00	2.6e-05	9.4e-04	7.8e-01
	STD	9.7e-07	6.1e-02	9.2e-11	3.3e-09	1.7e-06	2.8e-02	3.2e-02	3.2e-10	1.6e-08	**0.0e+00**	2.9e-03	3.2e+00	5.8e-05	1.0e-03	1.9e-01
F26	AVG	3.8e-06	2.8e+00	4.7e-10	5.6e-08	9.4e-06	1.3e-03	1.1e-03	6.7e-04	5.9e-02	**1.3e-32**	2.1e-02	4.2e-01	6.0e-03	1.7e-03	2.9e+00
	STD	4.8e-06	1.2e-01	3.7e-10	5.5e-08	9.69e-06	3.6e-03	3.3e-03	2.6e-03	4.1e-01	**2.7e-48**	9.1e-03	9.6e-01	3.0e-02	2.9e-03	5.7e-02
F27	AVG	**0.0e+00**	**0.0e+00**	1.1e+01	**0.0e+00**	**0.0e+00**	6.6e+00	1.1e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.0e-01	1.4e+01	**0.0e+00**	**0.0e+00**	1.4e-11
	STD	**0.0e+00**	**0.0e+00**	4.7e+00	**0.0e+00**	**0.0e+00**	2.5e+00	3.1e+00	**0.0e+00**	**0.0e+00**	**0.0e+00**	8.2e-02	5.7e+00	**0.0e+00**	**0.0e+00**	1.2e-12
F28	AVG	4.1e-23	9.7e-02	3.6e-01	2.3e-122	1.8e-53	4.1e-01	5.3e-01	1.3e-01	**0.0e+00**	**0.0e+00**	1.4e+00	6.3e-01	**0.0e+00**	9.1e-147	6.4e-07
	STD	2.9e-22	1.4e-02	5.5e-02	1.6e-121	7.3e-53	1.0e-01	1.1e-01	4.6e-02	**0.0e+00**	**0.0e+00**	3.3e-01	9.2e-02	**0.0e+00**	6.4e-146	3.7e-08
F29	AVG	1.8e-05	2.8e+00	4.4e-04	4.7e-08	4.6e-05	3.4e-03	3.1e-02	1.5e-03	1.4e-07	**1.3e-32**	1.6e+00	1.3e+06	3.3e-03	2.2e-02	2.9e+00
	STD	3.0e-05	6.0e-01	2.1e-03	5.7e-08	5.7e-05	1.4e-02	9.7e-02	4.3e-03	2.0e-07	**2.7e-48**	6.6e-01	6.5e+06	1.3e-02	2.2e-02	8.8e-02
F30	AVG	-4.4e+02	-2.3e+02	-1.9e+02	-4.4e+02	-4.4e+02	-4.0e+02	-3.8e+02	-3.7e+02	-4.4e+02	**-4.4e+02**	-4.4e+02	-1.4e+02	-2.1e+02	-4.4e+02	-1.4e+02
	STD	4.0e-03	1.6e+01	8.7e+00	2.6e+01	2.5e+01	6.5e+00	8.0e+00	1.8e+01	8.1e-04	**1.5e-13**	3.3e-02	5.2e+01	2.4e+01	1.9e-01	1.0e+00
F31	AVG	5.9e-39	**0.0e+00**	2.3e-03	1.8e-62	2.3e-28	5.0e-01	2.0e-01	1.6e-06	**0.0e+00**	**0.0e+00**	4.7e+00	1.0e+01	**0.0e+00**	4.7e-49	1.0e-02
	STD	1.4e-38	**0.0e+00**	4.2e-04	6.9e-62	7.9e-28	1.0e+00	4.6e-01	9.7e-07	**0.0e+00**	**0.0e+00**	4.0e-01	6.7e+00	**0.0e+00**	3.3e-48	3.0e-04
F32	AVG	**0.0e+00**	**0.0e+00**	3.0e-02	**0.0e+00**	**0.0e+00**	3.0e-02	3.8e-02	1.6e-03	**0.0e+00**	3.5e-03	7.0e-04	6.6e-02	**0.0e+00**	**0.0e+00**	4.9e-14
	STD	**0.0e+00**	**0.0e+00**	1.4e-02	**0.0e+00**	**0.0e+00**	1.2e-02	2.4e-02	1.0e-02	**0.0e+00**	2.5e-02	3.1e-04	1.0e-01	**0.0e+00**	**0.0e+00**	3.5e-15
F33	AVG	1.2e-10	**0.0e+00**	9.0e-10	5.5e-09	2.5e-12	2.3e-04	1.4e-04	4.0e-18	1.9e-28	9.0e-227	4.4e+02	1.3e+01	**0.0e+00**	3.6e-235	1.3e-11
	STD	5.2e-10	**0.0e+00**	2.3e-09	2.7e-08	1.7e-11	1.0e-03	5.6e-04	1.4e-17	1.4e-27	**0.0e+00**	1.7e+03	4.7e+01	**0.0e+00**	**0.0e+00**	1.3e-11
F34	AVG	1.0e-04	1.5e-01	**0.0e+00**	3.9e-31	8.2e-14	**0.0e+00**	**0.0e+00**	9.1e-22	4.0e-06	1.0e-30	8.4e-01	4.5e-02	**0.0e+00**	7.4e-10	7.3e-02
	STD	1.0e-04	3.2e-01	**0.0e+00**	9.0e-31	5.3e-13	**0.0e+00**	**0.0e+00**	2.5e-21	4.4e-06	7.2e-30	2.3e+00	1.8e-01	**0.0e+00**	1.4e-09	2.0e-01
F35	AVG	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.6e+00	2.5e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**
	STD	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.8e+00	4.0e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**
F36	AVG	-2.0e+00	-2.0e+00	**-2.0e+00**	**-2.0e+00**	-2.0e+00	**-2.0e+00**	**-2.0e+00**	**-2.0e+00**	-2.0e+00	**-2.0e+00**	-2.0e+00	-2.0e+00	-2.0e+00	-2.0e+00	-2.0e+00
	STD	1.4e-06	5.9e-07	8.9e-16	8.9e-16	9.4e-10	8.9e-16	8.9e-16	**5.6e-16**	2.5e-09	8.9e-16	4.4e-06	1.2e-05	2.1e-09	4.3e-11	3.7e-05
F37	AVG	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	-9.9e-01	**-1.0e+00**	**-1.0e+00**	-9.9e-01	-9.2e-01	-9.8e-01	**-1.0e+00**	**-1.0e+00**	-1.0e+00
	STD	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.2e-02	**0.0e+00**	**0.0e+00**	9.0e-03	7.7e-02	1.5e-02	**0.0e+00**	**0.0e+00**	3.8e-12
F38	AVG	1.1e+00	8.4e+00	**9.9e-01**	1.1e+00	1.4e+00	**9.9e-01**	1.1e+00	9.9e-01	9.9e-01	1.1e+00	9.9e-01	9.9e-01	9.9e-01	9.9e-01	2.0e+00
	STD	4.3e-01	4.5e+00	**0.0e+00**	1.3e+00	1.2e+00	**0.0e+00**	1.3e+00	1.2e-16	4.6e-11	1.1e+00	2.3e-10	3.6e-10	4.4e-11	3.5e-14	1.3e+00
F39	AVG	-3.8e+00	-3.8e+00	**-3.8e+00**	**-3.8e+00**	-3.8e+00	**-3.8e+00**	**-3.8e+00**	**-3.8e+00**	-3.8e+00	-3.8e+00	-3.5e+00	-3.8e+00	-3.8e+00	-3.8e+00	-3.8e+00
	STD	2.6e-03	4.2e-03	1.3e-15	1.3e-15	3.0e-04	1.3e-15	1.3e-15	**9.3e-16**	3.6e-03	3.1e-03	7.1e-01	4.0e-06	3.6e-06	4.5e-08	4.6e-02
F40	AVG	-3.2e+00	-3.1e+00	**-3.3e+00**	-3.2e+00	-3.1e+00	-3.3e+00	-3.3e+00	-3.3e+00	-3.1e+00	-3.3e+00	-3.2e+00	-3.2e+00	-3.3e+00	-3.2e+00	-2.7e+00
	STD	6.7e-02	5.0e-02	**8.9e-16**	6.7e-02	8.1e-02	4.1e-02	4.8e-02	2.4e-12	1.7e-01	5.6e-02	5.9e-02	5.9e-02	6.3e-04	4.6e-02	2.3e-01
F41	AVG	5.7e-03	2.5e-08	5.2e-109	9.6e-289	2.4e-04	7.7e-55	7.0e-103	1.0e-19	1.4e-06	2.2e-36	1.4e-04	1.3e-03	**0.0e+00**	1.3e-11	3.1e+00
	STD	5.7e-03	3.4e-08	2.2e-108	**0.0e+00**	6.2e-04	1.5e-54	2.8e-102	2.7e-19	1.9e-06	1.2e-35	1.5e-04	1.9e-03	**0.0e+00**	3.9e-11	2.3e+00
F42	AVG	-1.0e+00	-1.0e+00	**-1.0e+00**	**-1.0e+00**	-1.0e+00	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	-1.0e+00	**-1.0e+00**	-1.0e+00	-1.0e+00	-1.0e+00	-1.0e+00	-1.0e+00
	STD	1.3e-04	6.2e-08	**2.2e-16**	2.3e-16	1.2e-13	2.3e-16	**2.2e-16**	4.4e-16	9.4e-07	2.6e-16	4.1e-06	7.0e-06	3.6e-08	4.6e-11	2.1e-03
F43	AVG	4.6e-04	1.2e-02	6.5e-04	6.2e-04	3.8e-04	3.0e-04	**3.0e-04**	3.0e-04	3.5e-04	3.0e-04	1.2e-02	4.6e-03	3.4e-04	4.8e-04	1.0e-03
	STD	1.4e-04	2.2e-02	8.7e-05	2.5e-04	2.5e-04	1.3e-19	**1.1e-19**	2.2e-15	1.5e-04	1.3e-10	1.3e-02	7.9e-03	4.9e-05	2.5e-04	8.9e-04
F44	AVG	2.3e-158	**0.0e+00**	1.0e-43	4.9e-324	1.8e-135	2.6e-63	2.3e-299	2.9e-86	**0.0e+00**	**0.0e+00**	6.7e-05	4.0e-07	**0.0e+00**	**0.0e+00**	8.0e-17
	STD	1.6e-157	**0.0e+00**	5.4e-43	**0.0e+00**	1.0e-134	1.3e-62	**0.0e+00**	2.0e-85	**0.0e+00**	**0.0e+00**	4.9e-05	5.0e-07	**0.0e+00**	**0.0e+00**	6.3e-17
F45	AVG	7.0e-09	1.1e-04	5.0e-10	3.9e-13	5.0e-16	1.2e-100	**0.0e+00**	2.4e-35	1.5e-09	5.1e-09	2.1e-08	3.1e-13	5.4e-10	1.8e-10	8.3e-03
	STD	2.1e-08	2.5e-04	6.1e-10	3.4e-13	1.7e-15	9.0e-100	**0.0e+00**	7.7e-35	8.4e-10	3.0e-10	1.0e-08	7.7e-13	1.6e-09	1.6e-10	9.1e-03
F46	AVG	-1.0e+01	-3.7e+00	**-1.0e+01**	-1.0e+01	-5.4e+00	-1.0e+01	-9.9e+00	-1.0e+01	-1.0e+01	**-1.0e+01**	-5.2e+00	-5.1e+00	-1.0e+01	-1.0e+01	-4.4e+00
	STD	3.5e-03	1.0e+00	3.0e-15	7.2e-01	1.3e+00	7.1e-01	1.0e+00	2.0e-11	1.4e-04	**2.6e-15**	3.2e+00	3.1e+00	1.1e-06	1.1e-04	6.4e-01
F47	AVG	-1.0e+01	-4.3e+00	**-1.0e+01**	-1.0e+01	-5.4e+00	-1.0e+01	-9.9e+00	-1.0e+01	-1.0e+01	-1.0e+01	-5.3e+00	-5.5e+00	-1.0e+01	-1.0e+01	-4.5e+00
	STD	3.6e-03	1.3e+00	**1.7e-15**	1.0e+00	1.2e+00	9.4e-01	1.4e+00	1.9e-11	8.8e-05	7.5e-01	3.1e+00	3.4e+00	2.3e-05	8.6e-05	7.3e-01
F48	AVG	-1.0e+01	-3.9e+00	-1.0e+01	-1.0e+01	-5.4e+00	**-1.0e+01**	**-1.0e+01**	-1.0e+01	-1.0e+01	-1.0e+01	-4.9e+00	-5.5e+00	-1.0e+01	-1.0e+01	-4.5e+00
	STD	4.6e-03	1.0e+00	3.3e-15	2.8e-15	1.2e+00	**1.7e-15**	1.85e-15	1.4e-11	1.5e-04	2.8e-15	3.4e+00	3.4e+00	2.9e-05	1.1e-04	8.5e-01
F49	AVG	5.1e-02	1.2e-01	2.7e-01	4.3e-02	6.5e-02	2.2e-03	**2.2e-03**	2.2e-03	1.8e-02	9.4e-03	8.0e+00	1.3e-01	1.2e-02	3.1e-02	1.1e+00
	STD	3.7e-02	3.7e-01	3.2e-01	9.6e-02	6.6e-02	**1.1e-17**	1.2e-17	1.3e-10	9.4e-03	5.3e-03	1.2e+01	3.0e-01	6.8e-03	4.5e-02	9.4e-01
F50	AVG	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	8.5e-120	6.1e-119	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.7e-04	3.69e-04	**0.0e+00**	**0.0e+00**	1.5e-09
	STD	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.0e-119	2.6e-118	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.2e-04	3.69e-04	**0.0e+00**	**0.0e+00**	2.7e-10

rotated, composed and hybrid functions (Table 8). The implementation and properties of these functions can be found in https://github.com/P-N-Suganthan/2022-SO-BO.

These benchmarks offers a set of more challenging functions to be optimized. Therefore, for each algorithm, $(n_S) = 120$ individuals were used, meanwhile RECCA was executed with $n_S = 20$ and $n_{ne} = 6$. All algorithms applied $n_{it} = 2000$ iterations, and 30 independent runs were conducted for each algorithm, calculating the average value and standard deviation. Table 10 presents the obtained results, the best ones are shown in bold with an orange background.

RECAA obtained 3 of the best average values for the CEC 2022 functions, performing below the LSHADE and LSPACMA; and outperforming the other algorithms.

TABLE 6. Metaheuristics compared with RECAA on 500 and fixed dimensional problems.

Fn	Stats	AO	AOA	HBO	HGS	HHO	LSHADE	LSPACMA	MPA	MSCA	PO	PROS	PSS	RECAA	SMA	WSA
F1	AVG	2.1e-100	3.0e+17	3.5e+11	7.6e-230	6.5e-103	1.2e+03	1.1e+02	1.3e-18	**0.0e+00**	**0.0e+00**	2.4e+00	2.7e+04	**0.0e+00**	1.0e-309	1.7e-12
	STD	8.6e-100	2.7e+17	6.5e+11	**0.0e+00**	3.2e-102	1.9e+02	2.9e+01	1.1e-18	**0.0e+00**	**0.0e+00**	3.1e-01	1.6e+04	**0.0e+00**	**0.0e+00**	3.8e-14
F2	AVG	3.1e-216	3.5e-01	6.2e+09	**0.0e+00**	5.9e-201	1.0e+09	2.0e+08	4.0e-31	**0.0e+00**	**0.0e+00**	3.4e+06	3.2e+11	**0.0e+00**	**0.0e+00**	4.7e-19
	STD	**0.0e+00**	5.0e-02	9.9e+08	**0.0e+00**	**0.0e+00**	2.4e+08	6.6e+07	7.7e-31	**0.0e+00**	**0.0e+00**	6.9e+05	1.7e+10	**0.0e+00**	**0.0e+00**	2.0e-20
F3	AVG	-1.0e+00	-1.0e+00	-2.2e-02	-1.0e+00	-1.0e+00	-2.7e-01	-3.3e-01	-1.0e+00	-1.0e+00	-1.0e+00	-9.1e-01	**-5.9e-13**	-1.0e+00	-1.0e+00	-1.0e+00
	STD	**0.0e+00**	**0.0e+00**	7.3e-03	**0.0e+00**	**0.0e+00**	5.3e-02	4.6e-02	**0.0e+00**	**0.0e+00**	**0.0e+00**	8.2e-03	5.2e-13	**0.0e+00**	7.0e-04	5.8e-16
F4	AVG	8.4e-103	2.0e+03	1.0e+05	2.4e-236	1.7e-102	2.1e+03	2.9e+02	1.5e-17	**0.0e+00**	**0.0e+00**	4.5e+02	1.2e+05	**0.0e+00**	8.2e-199	3.6e-11
	STD	4.1e-102	2.3e+02	4.9e+04	**0.0e+00**	1.2e-101	3.7e+02	3.5e+01	1.2e-17	**0.0e+00**	**0.0e+00**	6.8e+01	7.3e+03	**0.0e+00**	**0.0e+00**	1.6e-12
F5	AVG	1.3e-103	8.4e+03	8.6e+04	6.6e-231	3.9e-101	1.5e+04	1.7e+03	1.5e-16	**0.0e+00**	**0.0e+00**	5.1e+02	6.69e+05	**0.0e+00**	2.5e-143	1.5e-10
	STD	5.8e-103	7.1e+02	9.69e+03	**0.0e+00**	2.4e-100	1.6e+03	3.3e+02	9.2e-17	**0.0e+00**	**0.0e+00**	4.0e+01	4.2e+04	**0.0e+00**	1.8e-142	3.0e-12
F6	AVG	2.1e-18	**0.0e+00**	5.5e-01	7.8e-97	1.2e-131	3.5e-18	1.6e-16	3.0e-60	**0.0e+00**	**0.0e+00**	3.3e-04	5.0e-06	**0.0e+00**	**0.0e+00**	4.6e-19
	STD	1.4e-17	**0.0e+00**	2.1e-01	5.4e-96	8.4e-131	1.7e-17	5.5e-16	1.3e-59	**0.0e+00**	**0.0e+00**	5.5e-04	1.6e-05	**0.0e+00**	**0.0e+00**	7.3e-19
F7	AVG	5.7e-317	3.5e-41	1.1e+03	**0.0e+00**	6.5e-199	6.4e+01	4.3e+01	2.5e-34	**0.0e+00**	**0.0e+00**	2.7e-01	9.9e+03	**0.0e+00**	**0.0e+00**	1.1e-26
	STD	**0.0e+00**	2.5e-40	1.5e+02	**0.0e+00**	**0.0e+00**	1.5e+01	7.5e+00	2.5e-34	**0.0e+00**	**0.0e+00**	1.2e-01	5.1e+02	**0.0e+00**	**0.0e+00**	4.6e-28
F8	AVG	-1.5e-01	-5.0e+00	8.9e+00	**-5.0e+00**	-2.5e+00	4.8e+00	-6.9e-01	-5.0e+00	-2e-162	-1.9e-06	-2.6e+00	3.3e+01	-1.5e+00	-1.6e+00	1.2e-06
	STD	4.5e-01	8.1e-05	5.1e-01	1.6e-08	2.4e+00	5.4e-01	2.7e-01	2.8e-03	**0.0e+00**	1.3e-05	2.2e-01	6.2e-01	2.2e+00	2.1e+00	1.6e-08
F9	AVG	4.9e-112	3.3e+01	8.9e+06	9.2e-01	4.3e-61	2.2e+05	1.3e+05	7.2e+03	**0.0e+00**	3.1e-249	2.6e+06	4.0e+06	8.5e-178	1.2e-37	1.4e-08
	STD	3.4e-111	2.7e+01	8.1e+05	6.5e+00	2.5e-60	3.8e+04	2.6e+04	5.4e+03	**0.0e+00**	**0.0e+00**	2.7e+05	4.4e+05	**0.0e+00**	8.2e-37	3.6e-09
F10	AVG	2.5e-63	7.9e+00	3.0e+03	1.3e-116	2.5e-52	3.0e+03	1.9e+03	7.1e-09	**0.0e+00**	4.2e-199	6.8e+02	1.2e+04	**0.0e+00**	3.7e-06	5.0e-04
	STD	1.7e-62	4.3e-01	1.2e+02	9.4e-116	1.4e-51	1.8e+02	1.5e+02	6.9e-09	**0.0e+00**	**0.0e+00**	3.2e+01	2.1e+02	**0.0e+00**	2.6e-05	6.9e-06
F11	AVG	4.0e-66	1.7e-01	9.8e+01	2.4e-109	5.9e-51	3.6e+01	3.0e+01	3.4e-05	**0.0e+00**	5.9e-170	8.7e+01	9.0e+01	**0.0e+00**	1.1e-31	2.1e-06
	STD	2.8e-65	1.5e-02	2.8e-01	1.7e-108	3.8e-50	1.8e+00	3.5e+00	2.3e-05	**0.0e+00**	**0.0e+00**	2.0e+00	2.5e+00	**0.0e+00**	8.0e-31	1.0e-08
F12	AVG	4.3e-77	7.9e+00	6.5e+203	2.1e+298	4.5e-51	3.6e+213	1.4e+174	5.1e+218	4.4e+193	1.7e-196	6.3e+208	9.1e+202	**0.0e+00**	3.9e-01	5.0e-04
	STD	2.8e-76	4.2e-01	3.2e-02	5.1e-03	2.7e-50	6.7e-08	6.2e-09	9.1e-05	9.2e-07	**0.0e+00**	3.9e-01	1.8e-09	**0.0e+00**	1.8e+00	6.8e-06
F13	AVG	**0.0e+00**	4.6e-08	8.5e+19	**0.0e+00**	**0.0e+00**	4.7e+15	2.7e+14	6.6e-59	**0.0e+00**	**0.0e+00**	3.1e+10	1.8e+20	**0.0e+00**	**0.0e+00**	6.9e-56
	STD	**0.0e+00**	2.6e-08	2.0e+19	**0.0e+00**	**0.0e+00**	2.2e+15	1.5e+14	2.7e-58	**0.0e+00**	**0.0e+00**	5.5e+10	2.2e+19	**0.0e+00**	**0.0e+00**	3.5e-57
F14	AVG	3.5e-153	5.9e-01	7.7e+04	3.0e-241	1.7e-101	3.2e+04	1.3e+04	5.9e-16	**0.0e+00**	**0.0e+00**	1.8e+03	5.69e+05	**0.0e+00**	8.0e-100	6.8e-10
	STD	2.4e-152	4.1e-02	6.0e+03	**0.0e+00**	1.1e-100	3.9e+03	3.3e+03	4.0e-16	**0.0e+00**	**0.0e+00**	1.7e+02	1.7e+04	**0.0e+00**	5.7e-99	1.3e-11
F15	AVG	**0.0e+00**	**0.0e+00**	7.6e+04	**0.0e+00**	**0.0e+00**	3.8e+04	1.9e+04	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.9e+03	5.7e+05	**0.0e+00**	**0.0e+00**	**0.0e+00**
	STD	**0.0e+00**	**0.0e+00**	6.9e+03	**0.0e+00**	**0.0e+00**	3.9e+03	4.2e+03	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.5e+02	1.7e+04	**0.0e+00**	**0.0e+00**	**0.0e+00**
F16	AVG	7.7e-148	1.3e+02	1.6e+07	1.2e-230	7.7e-97	6.7e+06	3.0e+06	1.1e-13	**0.0e+00**	**0.0e+00**	4.6e+05	1.2e+08	**0.0e+00**	1.4e-01	1.7e-07
	STD	5.4e-147	1.1e+01	1.3e+06	**0.0e+00**	5.4e-96	7.5e+05	6.0e+05	8.0e-14	**0.0e+00**	**0.0e+00**	5.1e+04	4.0e+06	**0.0e+00**	1.0e+00	3.5e-09
F17	AVG	**8.8e-16**	9.5e-03	1.9e+01	**8.8e-16**	**8.8e-16**	1.2e+01	8.5e+00	1.0e-09	**8.8e-16**	**8.8e-16**	4.3e+00	2.1e+01	**8.8e-16**	**8.8e-16**	1.6e-06
	STD	**0.0e+00**	4.1e-04	1.6e+00	**0.0e+00**	**0.0e+00**	3.9e-01	2.8e-01	3.7e-10	**0.0e+00**	**0.0e+00**	1.0e-01	1.9e-01	**0.0e+00**	**0.0e+00**	1.4e-08
F18	AVG	1.6e-04	1.1e-05	4.4e+02	2.0e-119	2.7e-53	1.3e+02	6.8e+01	1.5e-10	4.3e-07	4.0e-196	1.0e+01	7.0e+02	**0.0e+00**	2.8e-106	5.0e-06
	STD	5.7e-04	3.5e-05	4.2e+01	1.4e-118	1.2e-52	1.0e+01	8.3e+00	1.2e-10	2.3e-06	**0.0e+00**	9.69e-01	1.4e+01	**0.0e+00**	1.9e-105	6.3e-08
F19	AVG	-6e+203	-1.8e+96	-6.0e+57	-1e+224	-3e+223	-9e+148	-1e+115	-4e+202	-1e+224	-1e+224	-2e+204	-4e+114	**-2e+224**	-1e+224	-5e+22
	STD	1.7e+91	1.3e+97	2.7e+58	5.5e+26	8.5e+93	6.1e+149	8.1e+115	4.3e+73	6.5e+43	1.5e+81	2.6e+53	3.3e+115	3.7e+23	7.2e+47	**3.5e+23**
F20	AVG	2.7e-147	1.4e+04	3.5e+08	3.3e-240	9.0e-95	2.2e+08	1.4e+08	4.4e-12	**0.0e+00**	**0.0e+00**	1.4e+08	4.4e+09	**0.0e+00**	2.9e-01	4.1e-05
	STD	1.7e-146	2.5e+03	4.5e+07	**0.0e+00**	6.3e-94	3.9e+07	2.8e+07	3.3e-12	**0.0e+00**	**0.0e+00**	3.0e+07	3.4e+08	**0.0e+00**	2.0e+00	2.2e-06
F21	AVG	**0.0e+00**	**0.0e+00**	3.6e+01	**0.0e+00**	**0.0e+00**	2.6e+01	2.8e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.4e+00	1.0e+02	**0.0e+00**	**0.0e+00**	9.0e-13
	STD	**0.0e+00**	**0.0e+00**	2.0e+00	**0.0e+00**	**0.0e+00**	1.5e+00	1.5e+00	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.0e-01	1.8e+00	**0.0e+00**	**0.0e+00**	1.8e-14
F22	AVG	**0.0e+00**	2.6e-03	2.0e+01	**0.0e+00**	**0.0e+00**	9.1e+00	4.4e+00	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.4e+00	1.4e+02	**0.0e+00**	**0.0e+00**	3.7e-12
	STD	**0.0e+00**	2.4e-04	1.6e+00	**0.0e+00**	**0.0e+00**	9.2e-01	7.1e-01	**0.0e+00**	**0.0e+00**	**0.0e+00**	5.3e-02	3.2e+00	**0.0e+00**	**0.0e+00**	1.9e-13
F23	AVG	1.8e-02	4.5e+01	8.2e+02	4.8e-04	1.5e-04	1.4e+02	2.9e+01	3.3e+01	3.5e-04	**1.5e-32**	7.0e+00	1.8e+03	4.0e+01	4.4e-02	4.5e+01
	STD	2.5e-02	1.2e-01	1.0e+02	7.8e-04	2.5e-04	1.2e+01	3.8e+00	9.8e-01	2.4e-03	**0.0e+00**	5.1e-01	6.9e+01	2.8e+00	6.8e-02	1.6e-02
F24	AVG	1.1e-05	3.4e+01	1.7e+02	1.8e-03	2.6e-03	2.3e+02	2.2e+02	6.3e+01	5.9e-06	1.4e-02	1.1e+02	2.4e+02	**0.0e+00**	2.4e-03	1.5e+02
	STD	1.9e-05	4.4e+00	1.2e+01	4.4e-03	1.2e-02	8.1e-01	1.85e+00	3.2e+01	1.0e-05	5.6e-02	2.5e+00	7.6e-01	**0.0e+00**	5.6e-03	8.5e+01
F25	AVG	1.4e-07	1.0e+00	2.6e+08	7.6e-07	4.4e-07	9.1e+04	1.4e+01	2.8e-01	2.2e-09	**9.4e-34**	8.7e-01	2.4e+09	1.1e-01	1.4e-03	1.1e+00
	STD	1.8e-07	1.0e-02	3.9e+07	1.3e-06	5.9e-07	8.6e+04	8.3e+00	1.9e-02	3.6e-09	**1.7e-49**	1.1e-01	1.6e+08	4.7e-02	3.3e-03	2.3e-02
F26	AVG	4.5e-05	5.0e+01	5.6e+08	9.3e-01	7.9e-05	5.1e+06	8.2e+03	4.7e+01	6.2e-07	**1.3e-32**	1.4e+02	5.5e+09	4.8e+01	2.85e-01	5.0e+01
	STD	8.1e-05	3.6e-02	9.1e+07	6.6e+00	9.5e-05	1.8e+06	8.9e+03	4.2e-01	1.2e-06	**2.7e-48**	3.5e+02	3.5e+08	4.5e+00	7.4e-01	2.1e-03
F27	AVG	**0.0e+00**	2.1e-06	4.3e+03	**0.0e+00**	**0.0e+00**	3.1e+03	7.4e+02	**0.0e+00**	**0.0e+00**	**0.0e+00**	3.1e+02	5.4e+03	**0.0e+00**	**0.0e+00**	3.5e-10
	STD	**0.0e+00**	3.3e-06	3.5e+02	**0.0e+00**	**0.0e+00**	8.1e+02	6.6e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.3e+01	1.0e+02	**0.0e+00**	**0.0e+00**	9.1e-12
F28	AVG	3.9e-08	2.3e-01	3.6e+01	3.9e-110	2.8e-52	2.9e+01	1.6e+01	2.0e-01	**0.0e+00**	2.0e-03	3.2e+01	8.0e+01	**0.0e+00**	5.3e-37	2.8e-06
	STD	2.7e-07	6.5e-02	1.2e+00	2.8e-109	1.2e-51	1.3e+00	1.5e+00	8.2e-17	**0.0e+00**	1.4e-02	1.0e+00	1.4e+00	**0.0e+00**	3.7e-36	6.5e-08
F29	AVG	3.8e-04	3.1e+14	1.2e+13	1.6e-04	2.7e-04	5.6e+11	6.3e+11	4.7e+01	8.2e-07	**1.3e-32**	1.1e+09	1.0e+14	4.9e+01	1.1e+01	5.0e+01
	STD	6.9e-04	7.0e+13	1.9e+12	3.69e-04	5.0e-04	1.3e+11	1.8e+11	4.2e-01	1.5e-06	**2.7e-48**	4.8e+08	5.3e+12	1.0e+00	1.5e+01	2.5e-03
F30	AVG	-7.4e+03	-2.3e+03	-5.4e+02	-7.1e+03	-7.4e+03	-1.5e+03	-1.5e+03	-2.2e+03	-7.4e+03	**-7.4e+03**	-7.1e+03	-5.7e+02	-3.7e+03	-7.4e+03	-1.1e+03
	STD	7.5e-02	2.85e+01	3.1e+01	1.1e+03	1.4e+00	4.4e+01	4.9e+01	9.7e+01	2.2e-03	**8.3e-12**	2.7e+01	2.2e+02	6.3e+02	1.2e+01	6.5e+02
F31	AVG	3.9e-34	8.0e-09	9.5e+02	2.0e-58	5.5e-27	1.0e+03	7.4e+02	6.2e-05	**0.0e+00**	**0.0e+00**	4.3e+02	1.8e+03	**0.0e+00**	4.6e-01	2.1e-01
	STD	2.8e-33	4.0e-08	2.6e+01	1.1e-57	1.5e-26	2.4e+01	3.1e+01	5.6e-05	**0.0e+00**	**0.0e+00**	1.0e+01	1.2e+01	**0.0e+00**	2.6e+00	1.5e-03
F32	AVG	**0.0e+00**	3.8e-15	8.4e-01	**0.0e+00**	**0.0e+00**	7.7e-01	5.7e-01	**0.0e+00**	**0.0e+00**	7.1e-03	8.1e-02	7.0e-01	**0.0e+00**	**0.0e+00**	6.8e-14
	STD	**0.0e+00**	2.7e-14	1.4e-02	**0.0e+00**	**0.0e+00**	7.1e-03	1.5e-01	**0.0e+00**	**0.0e+00**	3.5e-02	4.3e-03	8.1e-03	**0.0e+00**	**0.0e+00**	1.4e-15
F33	AVG	1.2e-16	**0.0e+00**	1.85e+299	1.1e+305	3.8e-12	4.3e+161	3.7e+30	2.0e+157	5.5e+164	1.6e-226	2.8e+303	1.2e+227	**0.0e+00**	3.0e-05	1.2e-11
	STD	7.9e-16	**0.0e+00**	7.1e-26	2.1e-18	2.7e-11	1.4e-21	2.6e+31	6.2e-18	1.6e-17	**0.0e+00**	7.2e-13	1.5e-15	**0.0e+00**	1.4e-04	1.1e-11
F34	AVG	8.2e-05	1.3e-01	**0.0e+00**	3.7e-31	7.8e-14	**0.0e+00**	**0.0e+00**	1.0e-21	4.1e-06	6.1e-31	4.1e-01	3.0e-02	3.1e-07	7.6e-10	1.3e-02
	STD	1.1e-04	3.1e-01	**0.0e+00**	1.6e-30	3.7e-13	**0.0e+00**	**0.0e+00**	1.6e-21	6.1e-06	3.2e-30	5.9e-01	1.5e-01	2.2e-06	1.2e-09	4.3e-02
F35	AVG	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.1e+00	1.4e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**
	STD	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.85e+00	2.4e+01	**0.0e+00**	**0.0e+00**	**0.0e+00**
F36	AVG	-2.0e+00	-2.0e+00	**-2.0e+00**	**-2.0e+00**	-2.0e+00	**-2.0e+00**	**-2.0e+00**	**-2.0e+00**	-2.0e+00	**-2.0e+00**	-2.0e+00	-2.0e+00	-2.0e+00	-2.0e+00	-2.0e+00
	STD	6.1e-07	5.8e-07	8.9e-16	8.9e-16	5.9e-10	8.9e-16	8.9e-16	**5.4e-16**	4.9e-09	8.9e-16	4.0e-06	1.8e-05	2.6e-09	5.9e-11	3.9e-05
F37	AVG	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	-9.9e-01	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	-9.2e-01	-9.9e-01	**-1.0e+00**	**-1.0e+00**	-1.0e+00
	STD	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.5e-02	**0.0e+00**	**0.0e+00**	**0.0e+00**	6.7e-02	1.0e-02	**0.0e+00**	**0.0e+00**	2.9e-12
F38	AVG	1.2e+00	8.69e+00	**9.9e-01**	1.5e+00	1.2e+00	**9.9e-01**	**9.9e-01**	9.9e-01	9.9e-01	**9.9e-01**	9.9e-01	9.9e-01	9.9e-01	9.9e-01	2.2e+00
	STD	5.4e-01	4.2e+00	**0.0e+00**	2.3e+00	7.6e-01	**0.0e+00**	**0.0e+00**	1.4e-16	5.0e-11	**0.0e+00**	1.6e-10	3.9e-10	6.8e-12	9.4e-14	1.3e+00
F39	AVG	-3.8e+00	-3.8e+00	**-3.8e+00**	**-3.8e+00**	-3.8e+00	**-3.8e+00**	**-3.8e+00**	**-3.8e+00**	-3.8e+00	-3.8e+00	-3.1e+00	-3.8e+00	-3.8e+00	-3.8e+00	-3.8e+00
	STD	2.2e-03	3.6e-03	1.3e-15	1.3e-15	5.5e-04	1.3e-15	1.3e-15	**9.4e-16**	3.7e-03	3.3e-03	1.1e+00	4.0e-06	3.3e-06	3.9e-08	2.6e-02
F40	AVG	-3.2e+00	-3.1e+00	**-3.3e+00**	-3.2e+00	-3.1e+00	-3.3e+00	-3.2e+00	-3.3e+00	-3.1e+00	-3.2e+00	-3.2e+00	-3.2e+00	-3.3e+00	-3.2e+00	-2.7e+00
	STD	7.8e-02	7.4e-02	**9.0e-16**	6.5e-02	7.7e-02	4.1e-02	5.1e-02	4.5e-12	1.6e-01	8.1e-02	5.1e-02	5.9e-02	5.8e-04	4.9e-02	2.9e-01
F41	AVG	4.8e-03	4.3e-08	5.8e-108	1.8e-296	2.6e-04	6.7e-54	5.8e-103	5.2e-20	1.7e-06	9.8e-32	2.3e-04	1.5e-03	**0.0e+00**	6.7e-11	2.7e+00
	STD	3.6e-03	5.4e-08	3.8e-107	**0.0e+00**	6.7e-04	2.8e-53	1.8e-102	7.2e-20	1.8e-06	6.9e-31	4.5e-04	2.2e-03	**0.0e+00**	3.1e-10	2.4e+00
F42	AVG	-1.0e+00	-1.0e+00	**-1.0e+00**	**-1.0e+00**	-1.0e+00	**-1.0e+00**	**-1.0e+00**	**-1.0e+00**	-1.0e+00	**-1.0e+00**	-1.0e+00	-1.0e+00	-1.0e+00	-1.0e+00	-1.0e+00
	STD	1.9e-04	7.8e-08	2.3e-16	2.3e-16	1.9e-12	2.3e-16	**2.2e-16**	4.3e-16	1.0e-06	2.3e-16	2.4e-06	9.4e-06	3.9e-08	3.8e-11	4.7e-03
F43	AVG	4.5e-04	1.6e-02	6.5e-04	5.6e-04	3.69e-04	3.2e-04	**3.0e-04**	3.0e-04	3.2e-04	3.0e-04	1.3e-02	3.1e-03	3.9e-04	5.4e-04	9.8e-04
	STD	1.7e-04	3.1e-02	1.1e-04	2.0e-04	1.8e-04	1.2e-04	**9.4e-20**	1.9e-15	6.8e-05	1.4e-10	1.8e-02	6.4e-03	1.1e-04	2.7e-04	5.6e-04
F44	AVG	2.5e-153	**0.0e+00**	1.3e-43	5.3e-313	2.3e-141	2.6e-62	3.2e-297	3.3e-84	**0.0e+00**	**0.0e+00**	6.5e-05	2.7e-07	**0.0e+00**	**0.0e+00**	5.9e-17
	STD	1.6e-152	**0.0e+00**	6.6e-43	**0.0e+00**	1.6e-140	1.7e-61	**0.0e+00**	2.3e-83	**0.0e+00**	**0.0e+00**	5.0e-05	4.1e-07	**0.0e+00**	**0.0e+00**	7.2e-17
F45	AVG	4.8e-09	5.6e-05	5.8e-10	5.1e-13	7.1e-16	2.2e-109	**0.0e+00**	2.2e-35	1.6e-09	5.0e-09	2.6e-08	4.3e-13	5.1e-10	1.7e-10	1.4e-02
	STD	5.9e-09	1.5e-04	5.8e-10	5.3e-13	2.9e-15	1.2e-108	**0.0e+00**	5.3e-35	8.5e-10	3.9e-10	1.4e-08	1.1e-12	1.3e-09	1.6e-10	1.8e-02
F46	AVG	-1.0e+01	-3.9e+00	**-1.0e+01**	-1.0e+01	-5.5e+00	-1.0e+01	-9.9e+00	-1.0e+01	-1.0e+01	-1.0e+01	-5.7e+00	-5.0e+00	-1.0e+01	-1.0e+01	-4.6e+00
	STD	2.6e-03	1.0e+00	**2.6e-15**	2.5e-12	1.5e+00	7.1e-01	1.2e+00	1.0e-11	1.3e-04	7.1e-01	3.3e+00	2.9e+00	1.0e-06	7.7e-05	5.8e-01
F47	AVG	-1.0e+01	-4.2e+00	**-1.0e+01**	-1.0e+01	-5.6e+00	**-1.0e+01**	-1.0e+01	-1.0e+01	-1.0e+01	**-1.0e+01**	-5.3e+00	-6.3e+00	-1.0e+01	-1.0e+01	-4.6e+00
	STD	4.1e-03	1.4e+00	1.7e-15	7.5e-01	1.5e+00	**1.6e-15**	7.5e-01	2.8e-11	1.3e-04	1.6e-15	3.1e+00	3.4e+00	2.2e-05	7.9e-05	6.6e-01
F48	AVG	-1.0e+01	-3.9e+00	**-1.0e+01**	-1.0e+01	-5.8e+00	-1.0e+01	-1.0e+01	-1.0e+01	-1.0e+01	-1.0e+01	-4.0e+00	-5.5e+00	-1.0e+01	-1.0e+01	-4.5e+00
	STD	4.7e-03	1.4e+00	**1.5e-15**	2.7e-15	1.9e+00	3.5e-15	1.3e+00	2.8e-11	9.5e-05	3.0e-15	2.9e+00	3.3e+00	2.6e-05	1.3e-04	8.1e-01
F49	AVG	4.6e-02	3.4e-01	2.4e-01	3.4e-02	7.4e-02	2.2e-03	**2.2e-03**	2.2e-03	1.9e-02	1.1e-02	6.0e+00	6.5e-02	1.1e-02	2.0e-02	1.3e+00
	STD	3.0e-02	1.0e+00	1.6e-01	8.0e-02	9.2e-02	**1.0e-17**	1.1e-17	1.8e-10	2.3e-02	6.4e-03	8.8e+00	5.7e-02	5.2e-03	1.8e-02	1.1e+00
F50	AVG	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	9.9e-120	1.0e-118	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.4e-04	2.9e-04	**0.0e+00**	**0.0e+00**	1.4e-09
	STD	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	**0.0e+00**	2.2e-119	3.6e-118	**0.0e+00**	**0.0e+00**	**0.0e+00**	1.3e-04	2.9e-04	**0.0e+00**	**0.0e+00**	1.0e-10

RECAA generated 3 better standard deviation values, demonstrating an acceptable balance between exploration and exploitation. Table 9 shows the Wilcoxon rank-sum statistical. RECAA obtained the fourth-best rank behind LSHADE, MPA and very similar to LSPACMA. In general, RECAA obtained a positive difference for 11 of the 15 methods with respect to the number of functions with a better result in this experiment.

The proposed RECAA is simple and takes lesser execution time than almost all other algorithms for CEC 2022 functions. The average time of each algorithm was calculated for the 12 functions, and the results are illustrated in Fig. 8. It can be

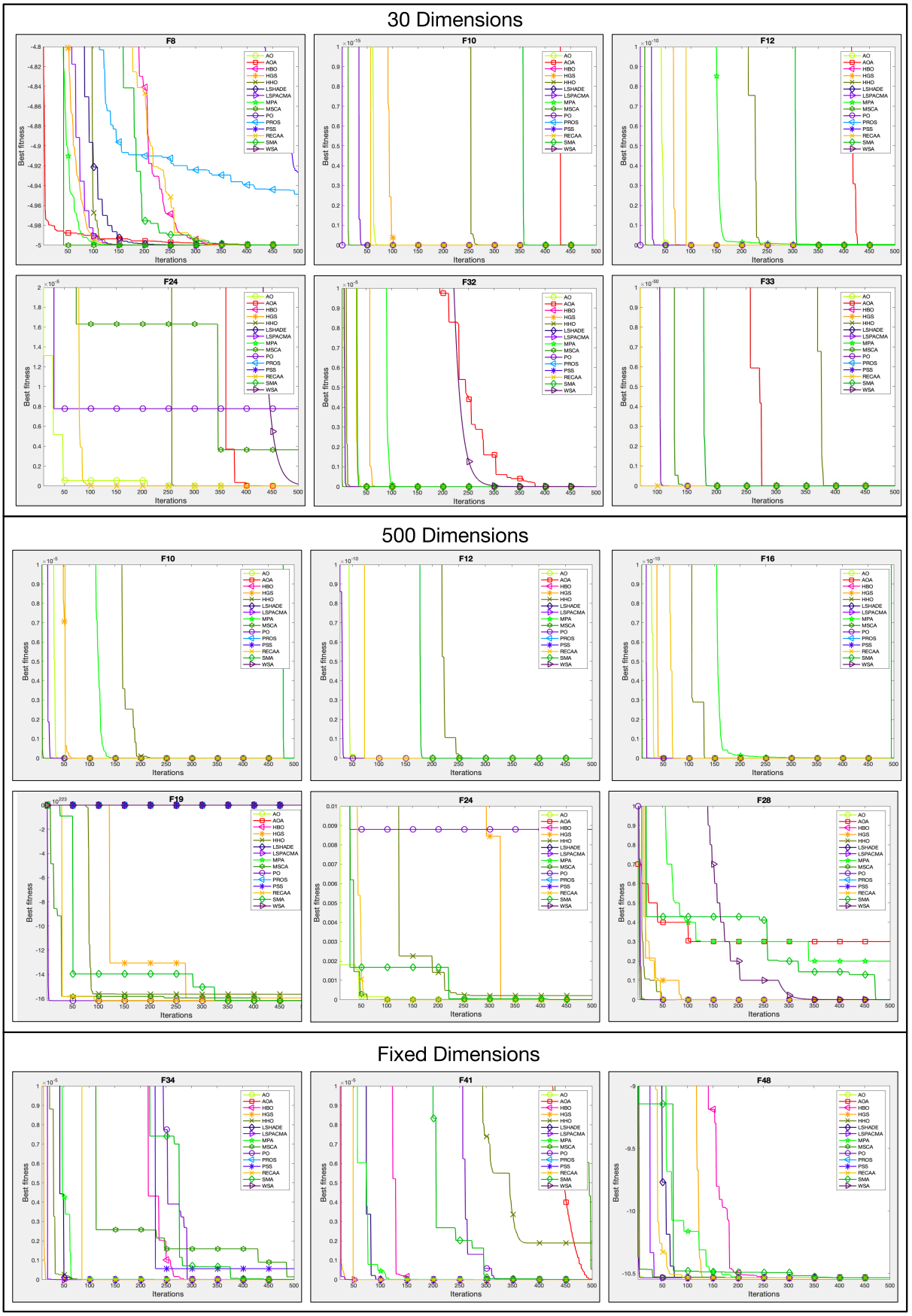

FIGURE 7. Convergence curves of the RECAA in several test functions.

TABLE 7. Wilcoxon rank-sum test and ranking of the compared algorithms on 500D and fixed problems.

Algorith	+/−/≈	Avg	Rank
AO	34/5/11	4.68	8
AOA	39/1/10	7.54	11
HBO	37/8/5	8.42	14
HGS	21/12/17	3.44	4
HHO	28/9/13	4.3	6
LSHADE	37/8/5	7.3	10
LSPACMA	39/8/3	6.7	9
MPA	21/11/18	4.46	7
MSCA	14/7/29	3.4	3
PO	13/13/24	2.48	1
PROS	45/3/2	8.4	13
PSS	46/1/3	10.44	15
RECAA	–/–/–	2.78	2
SMA	22/11/17	4.08	5
WSA	48/0/2	7.84	12

TABLE 8. CEC 2022 Single-objective bound-constrained benchmark functions.

No.	Name
F1	Shifted full rotated Zakharov function
F2	Shifted full rotated Rosenbrock's function
F3	Shifted full rotated expanded Schaffer's function
F4	Shifted full rotated non-continuous Rastrigin's function
F5	Shifted full rotated Levy function
F6	Hybrid Function 1
F7	Hybrid Function 2
F8	Hybrid Function 3
F9	Composition Function 1
F10	Composition Function 2
F11	Composition Function 3
F12	Composition Function 4

TABLE 9. Wilcoxon rank-sum test and ranking of the compared algorithms on CEC 2022 problems.

Algorith	+/−/≈	Avg	Rank
AO	10/0/2	8.83	9
AOA	12/0/0	12.67	13
HBO	7/3/2	5.42	5
HGS	8/3/1	6.67	7
HHO	11/1/0	9.67	11
LSHADE	2/9/1	2.67	1
LSPACMA	3/6/3	3.41	3
MPA	3/5/4	3.08	2
MSCA	12/0/0	10.42	12
PO	12/0/0	12.75	14
PROS	9/2/1	9.58	10
PSS	8/2/2	7.33	8
RECAA	–/–/–	3.52	4
SMA	7/2/3	5.67	6
WSA	12/0/0	14.17	15

seen that RECAA runs faster than 11 of the other 14 methods selected for comparison, with a run time very similar to AOA and slightly longer than LSHADE and WSA, demonstrating the competitiveness of RECAA in terms of execution time.

F. ANALYSIS OF CONVERGENCE OF THE RECAA

For a metaheuristic to posses an acceptable behavior, each individual must have abrupt position changes in the initial stages of the optimization process, indicating a good exploration process. These changes should be reduced as the process progresses and reaches its final stage to focus on exploiting the regions already detected as close to the optimal values (Van Den Bergh and Engelbrecht 2006). This dynamic can ensure that the metaheuristic converges to a position in the search space.

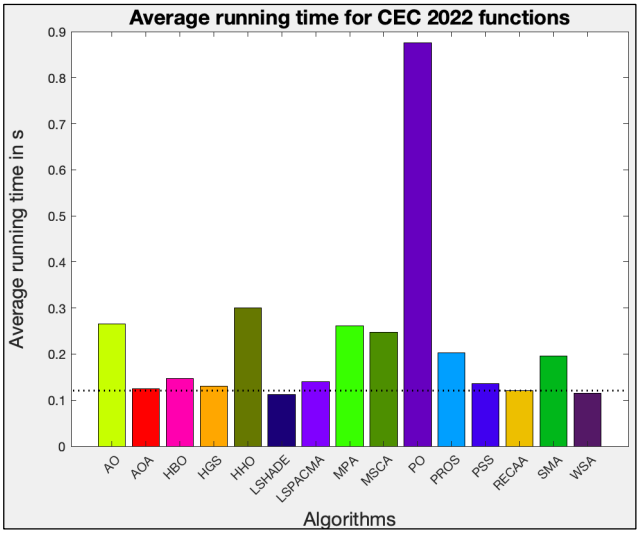

FIGURE 8. Average running time for the compared algorithms for CEC 2022 functions.

Fig. 9 shows the optimization process of RECAA on 9 test functions, the first 3 unimodal, the next 3 multimodal, both cases in 500 dimensions, and the last 3 with fixed dimensions. This experiment takes 50 independent runs of the RECAA with the parameters presented in Table 3 and the average values in the evolution of the 50 best smart-cells. This experiment was conducted to obtain the convergence and transition characteristics of the RECAA exploration-to-exploitation process.

In Fig. 9, the first column describes the function to be optimized in two dimensions. The second column shows the average movement of the first value of the best smart-cell in each iteration. The third column presents the average Euclidean distance between the previous and the current position of the best smart-cell in each iteration. The fourth column shows the weighted exploration-exploitation process as done in Fig. 6, while the last column presents the average convergence to the optimal value of the best smart-cell.

This experiment indicates that initially, RECAA performs an extensive search for optimal values in the search space characterized by the exploration rules and abrupt changes weighing more in the position of the smart-cells. These changes gradually decrease to favor the exploitation of promising areas in the search space, with minor changes in the last iterations due to the increased use of the exploitation rules. This experiment exemplifies the proper functioning of RECAA in the different test functions, where the convergence curves have an accelerated decay in the initial part of the process to prioritize the exploitation part at the end.

TABLE 10. Metaheuristics compared with RECAA on CEC 2022 problems.

Fn	Stats	AO	AOA	HBO	HGS	HHO	LSHADE	LSPACMA	MPA	MSCA	PO	PROS	PSS	RECAA	SMA	WSA
F1	AVG	8.81e+03	2.84e+04	2.00e+03	3.01e+02	3.07e+02	**3.00e+02**	2.23e+03	3.00e+02	6.02e+03	6.34e+04	5.00e+04	3.02e+02	3.52e+02	3.00e+02	2.82e+04
	STD	3.45e+03	9.27e+03	7.84e+02	4.39e+00	2.06e+00	**1.82e-14**	6.88e+03	1.56e-06	2.90e+03	1.01e+04	1.91e+04	7.15e+00	2.89e+01	3.04e-03	4.28e+03
F2	AVG	4.63e+02	1.27e+03	4.49e+02	4.45e+02	4.55e+02	4.49e+02	4.49e+02	4.34e+02	6.14e+02	6.37e+02	4.50e+02	4.48e+02	**4.23e+02**	4.46e+02	2.49e+03
	STD	1.62e+01	3.52e+02	1.65e+01	1.70e+01	1.87e+01	1.27e+00	1.27e+00	1.98e+01	9.19e+01	6.35e+01	1.96e+01	2.04e+01	**2.95e-01**	1.36e+01	4.46e+02
F3	AVG	6.28e+02	6.53e+02	**6.00e+02**	6.00e+02	6.50e+02	6.00e+02	6.00e+02	6.02e+02	6.33e+02	6.43e+02	6.00e+02	6.02e+02	6.00e+02	6.00e+02	6.78e+02
	STD	6.57e+00	7.01e+00	**1.13e-13**	5.03e-01	9.60e+00	1.13e-06	1.16e-07	1.28e-02	1.00e+01	6.38e+00	1.25e-02	1.46e+00	1.09e+00	1.17e-01	6.40e+00
F4	AVG	8.66e+02	8.97e+02	9.00e+02	8.97e+02	8.80e+02	8.07e+02	**8.06e+02**	8.28e+02	9.03e+02	9.06e+02	9.78e+02	9.22e+02	8.65e+02	8.64e+02	9.75e+02
	STD	1.54e+01	2.20e+01	7.67e+00	2.52e+01	1.43e+01	**1.75e+00**	2.51e+00	6.05e+00	1.69e+01	1.64e+01	5.98e+01	3.08e+01	8.86e+00	2.34e+01	9.45e+00
F5	AVG	1.81e+03	2.53e+03	1.06e+03	2.18e+03	2.39e+03	**9.00e+02**	**9.00e+02**	9.02e+02	2.10e+03	2.45e+03	5.24e+03	2.04e+03	9.00e+02	1.18e+03	3.30e+03
	STD	3.50e+02	3.38e+02	2.15e-02	5.32e+02	2.98e+02	**0.00e+00**	**0.00e+00**	1.09e+00	4.92e+02	4.56e+02	2.09e+03	1.29e+03	2.44e+00	5.88e+02	3.57e+02
F6	AVG	2.21e+04	1.81e+05	2.08e+05	1.22e+04	3.15e+04	1.81e+03	1.84e+03	**1.80e+03**	6.10e+06	2.05e+08	8.68e+03	2.73e+05	2.31e+03	2.27e+04	1.00e+08
	STD	1.06e+04	1.23e+06	1.42e+05	8.91e+03	2.18e+04	7.09e+00	2.18e+01	1.45e+00	5.63e+06	1.87e+08	6.21e+03	4.19e+05	2.83e+02	4.73e+03	**0.00e+00**
F7	AVG	2.08e+03	2.18e+03	2.04e+03	2.06e+03	2.15e+03	**2.02e+03**	2.02e+03	2.03e+03	2.09e+03	2.16e+03	2.14e+03	2.06e+03	2.03e+03	2.04e+03	2.22e+03
	STD	2.27e+01	5.18e+01	4.87e+00	3.00e+01	5.55e+01	6.35e+00	**2.51e+00**	3.95e+00	2.50e+01	2.37e+01	9.66e+01	2.89e+01	4.28e+00	2.35e+01	2.91e+01
F8	AVG	2.23e+03	2.36e+03	2.23e+03	2.23e+03	2.25e+03	2.22e+03	**2.22e+03**	2.22e+03	2.24e+03	2.28e+03	2.28e+03	2.23e+03	2.23e+03	2.23e+03	2.54e+03
	STD	1.03e+01	1.17e+02	7.70e-01	2.88e+01	3.03e+01	**5.98e-01**	5.44e+00	3.47e+00	7.07e+00	2.17e+01	6.96e+01	4.88e+00	1.27e+00	2.40e+01	2.29e+02
F9	AVG	2.49e+03	2.72e+03	2.48e+03	2.48e+03	2.48e+03	2.48e+03	2.48e+03	2.48e+03	2.52e+03	2.66e+03	2.49e+03	2.48e+03	**2.48e+03**	2.48e+03	3.26e+03
	STD	7.03e+00	1.01e+02	6.19e-13	6.67e-03	6.89e-03	6.03e-13	5.14e-13	4.16e-08	2.01e+01	1.21e+02	2.08e+01	6.03e-01	**4.59e-13**	1.17e-02	2.09e+02
F10	AVG	2.57e+03	4.27e+03	2.48e+03	2.55e+03	3.16e+03	2.50e+03	2.50e+03	2.50e+03	3.04e+03	4.01e+03	**2.41e+03**	2.47e+03	2.57e+03	2.71e+03	5.51e+03
	STD	2.67e+02	9.91e+02	2.00e+01	8.66e+01	6.99e+02	2.71e+01	1.59e+01	**7.85e-02**	1.02e+03	1.79e+03	4.42e+01	7.75e+01	1.84e+02	1.92e+02	1.07e+03
F11	AVG	2.97e+03	6.96e+03	2.90e+03	2.93e+03	2.95e+03	2.91e+03	2.92e+03	2.81e+03	4.31e+03	4.67e+03	3.02e+03	2.95e+03	**2.67e+03**	2.93e+03	8.49e+03
	STD	9.43e+01	7.48e+02	1.47e+00	6.88e+01	1.06e+02	2.40e+01	3.88e+01	1.35e+02	6.77e+02	5.70e+02	4.51e+02	7.56e+01	**4.16e-13**	8.39e+01	5.06e+02
F12	AVG	2.98e+03	3.52e+03	2.94e+03	2.95e+03	3.03e+03	2.94e+03	2.94e+03	**2.93e+03**	2.97e+03	3.04e+03	2.99e+03	2.97e+03	2.95e+03	2.94e+03	3.86e+03
	STD	1.82e+01	1.79e+02	**1.47e+00**	1.30e+01	5.25e+01	3.73e+00	4.77e+00	2.26e+00	1.48e+01	2.56e+01	2.80e+01	2.04e+01	5.00e+00	5.28e+00	1.73e+02

One of the reasons for the superior performance of the RECAA is the application of a cellular automaton-type neighborhood, where concurrent exploration and exploitation actions are used, and the best neighbor is chosen as the new smart-cell.

This simultaneous action of the rules to obtain new solutions allows a fast convergence in the early stages of optimization. For multimodal functions, the performance of the RECAA is similar to the PO and superior to the rest of the algorithms.

The random behavior in the concurrence of rules allows RECAA as well to obtain a good performance in the exploitation process. This feature can be seen in the results of RECAA for unimodal functions, where it performs similarly to MSCA and is superior to the rest of the algorithms.

The same concurrence of rules allows RECAA to escape local minima and obtain better optimal values. This property can be observed in the fixed-dimensional functions, where RECAA obtains consistent results that are better than most of the algorithms taken for comparison.

In summary, the application of a neighborhood with simultaneous exploration and exploitation actions, elitism, the random component in the application of the rules, and the selection of the best neighbor to update each smart-cell are critical factors for a good performance in both exploration and exploitation actions. Finally, the simplicity of the rules permits the utilization of RECAA for increasing dimensional problems, making it attractive for applications in various areas.

In the next section, RECAA is applied to several engineering problems to demonstrate its performance.

VI. APPLICATION IN ENGINEERING PROBLEMS

This section reviews three engineering problems treated in recent works and characterized by their complexity, dimensionality, and constraints that must be met for their solution.

RECAA employs a scalar penalty function for constraint handling, where solutions that do not meet the constraints are penalized by considerably increasing their fitness value, for instance, adding a value of 10000 for each constraint violation. This approach is easily implemented and offers good results without implying an increment of the computational complexity [48]. For each problem, 30 independent runs were taken to compare the best results against those published in recent literature.

A. TENSION/COMPRESSION SPRING DESIGN

In this first engineering case, the objective is to optimize the design of a tension/compression spring by minimizing its weight.

The problem has 3 variables $\boldsymbol{y} = (y(1), y(2), y(3))$ to calculate the weight; the wire diameter $y(1)$, the average coil diameter $y(2)$ and the number of coils $y(3)$ [44], [49]. Figure 10 presents the description of these variables. The problem also has 4 constraints, and its mathematical formulation is presented in Eq. 1.

$$\min f(\boldsymbol{y}) = y(1)^2 y(2)(2 + y(3))$$

Subject to:

$$g_1(\boldsymbol{y}) = 1 - \frac{y(2)^3 y(3)}{71785 y(1)^4} \leq 0$$

$$g_2(\boldsymbol{y}) = \frac{4y(2)^2 - y(1)y(2)}{12566 y(2) y(1)^3 - y(1)^4} + \frac{1}{5108 y(1)^2} - 1 \leq 0$$

$$g_3(\boldsymbol{y}) = 1 - \frac{140.45\, y(1)}{y(2)^2 y(3)} \leq 0$$

$$g_4(\boldsymbol{y}) = \frac{y(1) + y(2)}{1.5} - 1 \leq 0$$

where :

$$0.05 \leq y(1) \leq 2.0$$
$$0.25 \leq y(2) \leq 1.3$$
$$2.00 \leq y(3) \leq 15.0 \qquad (1)$$

For this problem, the RECAA is compared to the published results using the war strategy optimization [31], the Nelder-Mead simplex search and particle swarm optimization (NM-PSO) [50], the political optimize (PO) [44], the differential evolution with dynamic stochastic selection (DEDS) [51], the hybrid evolutionary algorithm and adaptive constraint-handling technique (HEAA) [52], the differential evolution with level comparison (DELC) [53], the water cycle

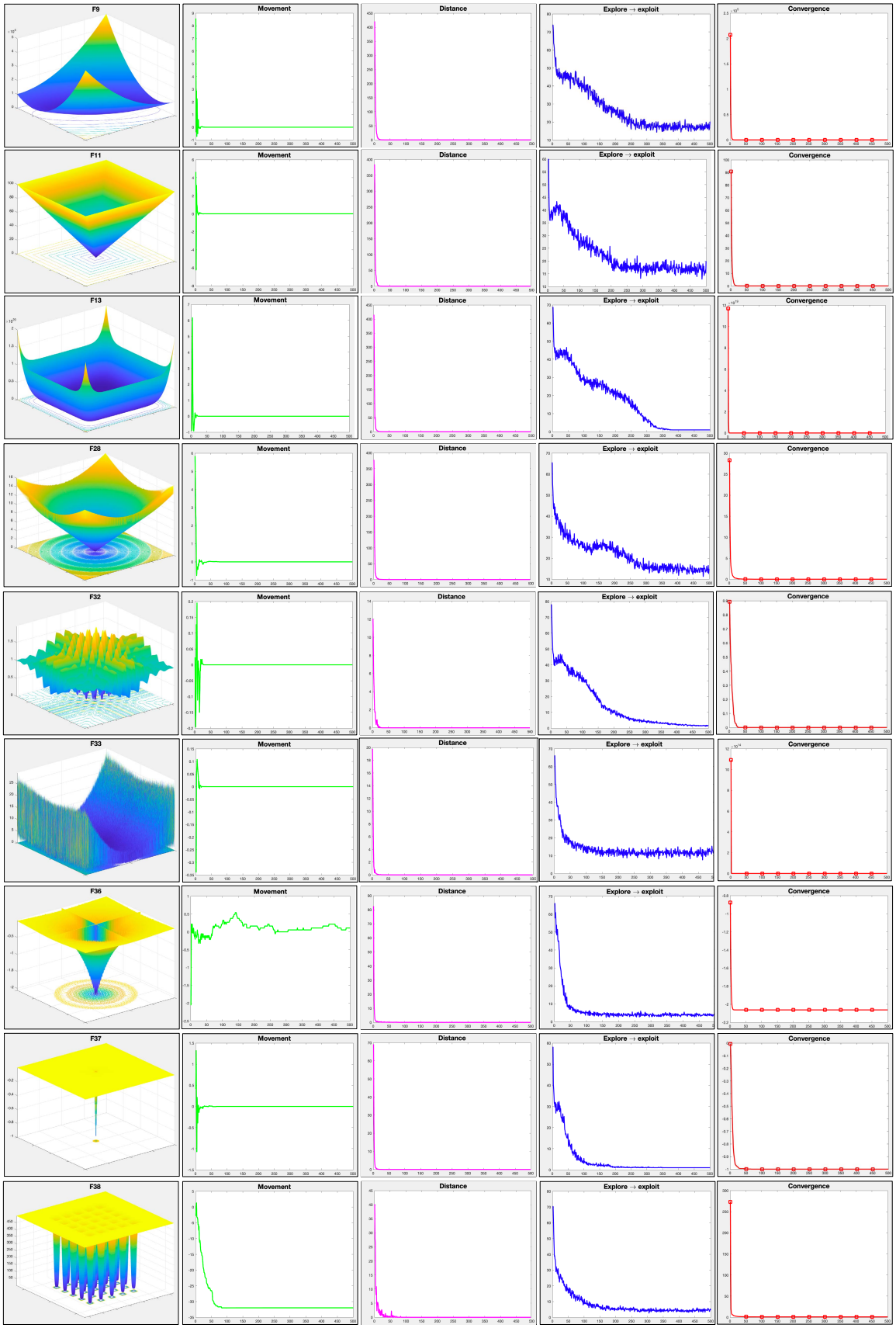

FIGURE 9. Average of movement, distance, exploration-exploitation weighting, and convergence of the best smart-cell in the RECAA.

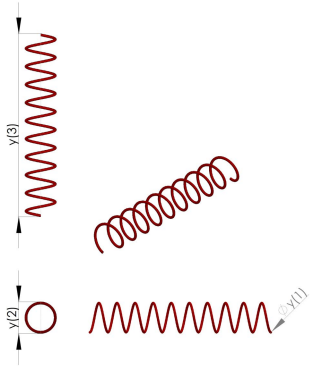

FIGURE 10. Tension/compression spring design.

TABLE 11. Comparison of the best solutions obtained for the tension/compression spring design problem.

Algorithm	$y(1)$	$y(2)$	$y(3)$	$f(\boldsymbol{y})$
WSO	0.05	0.3489	10.5622	0.01095
NM–PSO	0.05162	0.3550	11.3327	0.01263
RECAA	0.0518252	0.3	11.099148	0.012666
PO	0.05248	0.3594	10.209	0.01267
DEDS	0.05169	0.32	11.2897	0.01267
HEAA	0.05169	0.33	11.2829	0.01267
DELC	0.05169	0.32	11.2897	0.01267
WCA	0.05168	0.352	11.3041	0.01267
MADE	0.05169	0.32	11.2897	0.01267

algorithm (WCA) [54], and the modified adaptive differential evolution (MADE) [55].

For the operation of RECAA, the parameters indicated in Table 3 were considered, except for the number of iterations, which was taken as $n_{it} = 50$ to be in correspondence with the experiment performed in [44]. The results obtained by the RECAA and their comparison with the other algorithms are presented in Table 11

The results show that RECAA obtains the third-best result among the compared algorithms, very close to NM-PSO. It is worth mentioning that the solution found by the NM-PSO does not meet some of the constraints of Eq. 1, as pointed out in [44]. Thus, RECAA finds a solution that addresses all the constraints of the problem and is comparable to those obtained by recent and well-tested algorithms.

B. MULTIPLE DISK CLUTCH BRAKE PROBLEM

The optimization of this problem needs to find discrete values. For this reason, a version of RECAA was used that simply rounds the values of each individual after applying the evolution rules. The objective is to minimize the mass of the multiple disk clutch brake, which depends on a vector $\boldsymbol{y}$ of 5 variables, the inner radius $y(1)$, the outer radius $y(2)$, the disk thickness $y(3)$, the actuator force $y(4)$, and the number of surfaces in friction $y(5)$. Figure 11 describes this system.

The problem contemplates 8 nonlinear constraints. The mathematical model is presented in Eq. 2.

$$min\, f(\boldsymbol{y}) = \pi(y(2)^2 - y(1)^2)y(3)(y(5)+1)\rho$$

Subject to:

$$g_1(\boldsymbol{y}) = y(2) - y(1) - \Delta r \geq 0$$

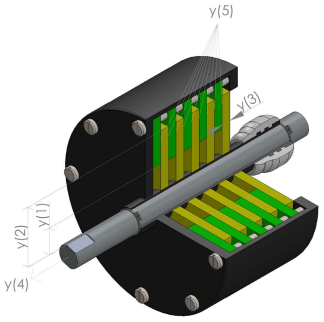

FIGURE 11. Multiple disk clutch brake design.

TABLE 12. Comparison of the best solutions obtained for the multiple disk clutch brake problem.

Algorithm	$y(1)$	$y(2)$	$y(3)$	$y(4)$	$y(5)$	$f(\boldsymbol{y})$
RECAA	70	90	1	792	2	0.2242
TLBO	70	90	1	810	3	0.356
WCA	70	90	1	910	3	0.356
HGS	70	90	1	1000	3	0.357
PVS	70	90	1	980	3	0.357

$$g_2(\boldsymbol{y}) = L_{max} - (y(5)+1)(y(3)+\delta) \geq 0$$
$$g_3(\boldsymbol{y}) = P_{max} - P_{rz} \geq 0 \quad g_4(\boldsymbol{y}) = P_{max} v_{sr\ max} - P_{rz} v_{sr} \geq 0$$
$$g_5(\boldsymbol{y}) = v_{sr\ max} - v_{sr} \geq 0 \quad g_6(\boldsymbol{y}) = T_{max} - T \geq 0$$
$$g_7(\boldsymbol{y}) = M_h - sM_s \geq 0 \quad g_8(\boldsymbol{y}) = T \geq 0$$

where:

$$\rho = 0.0000078 kg/mm^3 \quad \Delta r = 20\ mm \quad L_{max} = 30mm$$
$$\delta = 0.5 \quad P_{max} = 1MPa \quad P_{rz} = \frac{y(4)}{A} N/mm^2$$
$$A = \pi(y(2)^2 - y(1)^2) mm^2 \quad v_{sr\ max} = 10m/s$$
$$v_{sr} = \frac{\pi R_{sr} n}{30} mm/s$$
$$R_{sr} = \frac{2}{3}\frac{y(2)^3 - y(1)^3}{y(2)^2 y(1)^2} mm \quad n = 250rpm\ T_{max} = 15s$$
$$T = \frac{I_z \pi n}{30(M_h + M_f)} mm \quad I_z = 55 kg/m^2$$
$$M_h = \frac{2}{3}\mu y(4) y(5) \frac{y(2)^3 - y(1)^3}{y(2)^2 - y(1)^2}$$
$$M_f = 3Nm\ \mu = 0.5\ s = 1.5\ M_s = 40Nm$$
$$60 \leq y(1) \leq 80\ 90 \leq y(2) \leq 110\ 1 \leq y(3) \leq 3$$
$$0 \leq y(4) \leq 1000\ 2 \leq y(5) \leq 9 \qquad (2)$$

In this case, the published results using the teaching-learning-based optimization (TLBO) [56], the water cycle algorithm (WCA) [54], the passing vehicle search (PVS) [57] and the hunger games search (HGS) [27] are taken as a comparison. RECAA again uses the parameters indicated in Table 3, with a number of iterations $n_{it} = 50$ to correspond with the experiments performed on the algorithms taken as reference. The comparative results obtained by the RECAA are shown in Table 12.

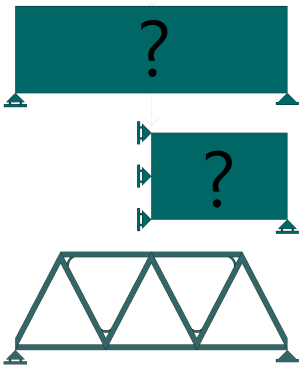

FIGURE 12. Scheme of the topology optimization.

In Table 12 RECAA obtains the best result among the compared algorithms, fulfilling all the problem's constraints. This result shows that RECAA is suitable for optimizing the multiple disk clutch brake problem.

C. TOPOLOGY OPTIMIZATION

This problem involves optimizing the material distribution to build a structural support element where a predefined set of loads is considered. In this problem, we have a vector $\boldsymbol{y}$ with $N = 30$ design variables to specify the structure's geometry, with N inequalities of design constraints. Figure 12 gives a scheme of this case.

A formulation of the problem can be found in [58]; the mathematical model is presented in Eq. 3.

$$
\begin{aligned}
&min\, f(\boldsymbol{y}) = \boldsymbol{u}^T \boldsymbol{K}\boldsymbol{u} = \sum_{i=1}^{N} (y(i))^p \boldsymbol{u}_i^T \boldsymbol{K}\boldsymbol{u}_i \\
&\text{Subject to:} \\
&\quad g_1(\boldsymbol{y}) = \frac{V(\boldsymbol{y})}{V_0} - h = 0 \\
&\quad g_2(\boldsymbol{y}) = \boldsymbol{K}\boldsymbol{u} - \boldsymbol{h} = \boldsymbol{0} \\
&\text{where:} \\
&\quad \boldsymbol{0} \leq \boldsymbol{y}_{\boldsymbol{min}} \leq \boldsymbol{y} \leq \boldsymbol{1}
\end{aligned}
\tag{3}
$$

where $\boldsymbol{h}$ is the force vector, $\boldsymbol{u}$ is the global displacement vector, $\boldsymbol{K}$ is the global stiffness matrix, $\boldsymbol{u}_i$ are the elements of each vector, $\boldsymbol{y}_{min}$ is a positive vector of minimum densities, $p = 3$ is a penalty power, $V(\boldsymbol{y})$ and V_0 represent the material volume and the design volume, and h is the prescribed volume fraction.

Here, we compare the published results using the improved unified differential evolution algorithm (UIDE), the matrix adaptation evolution strategy (MAES), the linear success-history-based adaptive differential evolution (LSHADE), and the atomic orbital search (AOS) [31]. RECAA again takes the parameters from Table 3 with $n_{it} = 50$ iterations to fit with the algorithms taken as reference. The comparative results are presented in Table 13.

TABLE 13. Comparison of the best solutions for the topology optimization problem.

Algorithm	$f(\boldsymbol{y})$
AOS	2.639346497
RECAA	2.639346497
UIDE	2.64
LSHADE	2.64
MAES	2.65

The results show that RECAA calculates a result almost identical to AOS and is competitive with those obtained by algorithms well-known for their excellent performance.

VII. CONCLUSION AND FURTHER WORK

This paper presents a new global optimization algorithm called RECAA inspired by evolution, neighborhood and local interaction rules of reversible elementary cellular automata. The randomness, concurrency, and information exchange between the smart-cells generated by applying the different rules, produce an appropriate balance between exploration and exploitation actions during the optimization process.

Parameter tuning and comparative computational testing was done with 50 test functions, 16 unimodal and 17 multimodal functions in 30 and 500 dimensions, plus 17 fixed-dimensional functions. The three groups were used to evaluate RECAA and its balance between exploration and exploitation and were compared against other 14 recently published algorithms recognized for their efficiency. The experiments showed satisfactory performance of RECAA.

Engineering problems used in recent literature were also taken to test RECAA against results obtained by other recent methods. RECAA also demonstrated its high quality in finding solutions to these problems, proving, in general, its competitiveness against other recent metaheuristics.

Proposed future work is to continue drawing inspiration from other dynamic behaviors of cellular automata such as the complexity of well-known rules like Rule 54, Rule 110, or LIFE, the non-trivial collective behavior of other types of automata such as reaction-diffusion, traffic, memory usage, lattice gases, among others to establish new metaheuristic algorithms. Another line of research is to adapt RECAA with multi-objective strategies to address a more extensive number of problems. applications and extensions.

REFERENCES

[1] R. Sarker, J. Kamruzzaman, and C. Newton, "Evolutionary optimization (EvOpt): A brief review and analysis," *Int. J. Comput. Intell. Appl.*, vol. 3, no. 4, pp. 311–330, Dec. 2003.
[2] K. Kumar and J. Paulo Davim, *Optimization Using Evolutionary Algorithms and Metaheuristics: Applications in Engineering*, 1 ed. Boca Raton, FL, USA: CRC Press, 2020.
[3] A. Gogna and A. Tayal, "Metaheuristics: Review and application," *J. Exp. Theor. Artif. Intell.*, vol. 25, no. 4, pp. 503–526, 2013.
[4] A. E. Hassanien and E. Emary, *Swarm Intelligence: Principles, Advances, and Applications*. Boca Raton, FL, USA: CRC Press, 2018.
[5] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[6] H. V. McIntosh, *One Dimensional Cellular Automata*. London, U.K.: Luniver Press, 2009.

[7] S. M. Bilan, M. M. Bilan, and R. L. Motornyuk, *New Methods and Paradigms for Modeling Dynamic Processes Based on Cellular Automata*. Hershey, PA, USA: IGI Global, 2020.
[8] J. Kari, "Reversible cellular automata: From fundamental classical results to recent developments," *New Gener. Comput.*, vol. 36, no. 3, pp. 145–172, Jul. 2018.
[9] J. C. Seck-Tuoh-Mora, N. Hernandez-Romero, P. Lagos-Eulogio, J. Medina-Marin, and N. S. Zuñiga-Peña, "A continuous-state cellular automata algorithm for global optimization," *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114930.
[10] J. C. Seck-Tuoh-Mora, N. Hernandez-Romero, F. Santander-Baños, V. Volpi-Leon, J. Medina-Marin, and P. Lagos-Eulogio, "A majority–minority cellular automata algorithm for global optimization," *Expert Syst. Appl.*, vol. 203, Oct. 2022, Art. no. 117379.
[11] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.
[12] S. Akyol and B. Alatas, "Plant intelligence based metaheuristic optimization algorithms," *Artif. Intell. Rev.*, vol. 47, no. 4, pp. 417–462, 2017.
[13] B. Alatas and H. Bingol, "Comparative assessment of light-based intelligent search and optimization algorithms," *Light Eng.*, vol. 28, no. 6, pp. 51–59, 2020.
[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.
[15] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
[16] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in *Proc. Int. Fuzzy Syst. Assoc. World Congr.* Berlin, Germany: Springer, 2007, pp. 789–798.
[17] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.
[18] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, Feb. 2016.
[19] Y. Ling, Y. Zhou, and Q. Luo, "Lévy flight trajectory-based whale optimization algorithm for global optimization," *IEEE Access*, vol. 5, pp. 6168–6186, 2017.
[20] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, 2011.
[21] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[22] C. Li, J. Li, H. Chen, M. Jin, and H. Ren, "Enhanced Harris hawks optimization with multi-strategy for global optimization tasks," *Expert Syst. Appl.*, vol. 185, Dec. 2021, Art. no. 115499.
[23] B. Xing and W.-J. Gao, "Fruit fly optimization algorithm," in *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms*. Cham, Switzerland: Springer, 2014, pp. 167–170.
[24] Y. Fan, P. Wang, A. A. Heidari, M. Wang, X. Zhao, H. Chen, and C. Li, "Boosted hunting-based fruit fly optimization and advances in real-world problems," *Expert Syst. Appl.*, vol. 159, Nov. 2020, Art. no. 113502.
[25] G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, pp. 1995–2014, Jul. 2019.
[26] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization," *Future Gener. Comput. Syst.*, vol. 111, pp. 300–323, Oct. 2020.
[27] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, "Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114864.
[28] I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, and H. Chen, "RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method," *Expert Syst. Appl.*, vol. 181, Nov. 2021, Art. no. 115079.
[29] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.
[30] S. Gupta, K. Deep, S. Mirjalili, and J. H. Kim, "A modified sine cosine algorithm with novel transition parameter and mutation operator for global optimization," *Expert Syst. Appl.*, vol. 154, Sep. 2020, Art. no. 113395.
[31] M. Azizi, S. Talatahari, and A. Giaralis, "Optimization of engineering design problems using atomic orbital search algorithm," *IEEE Access*, vol. 9, pp. 102497–102519, 2021.
[32] J. C. Seck-Tuoh-Mora, J. Medina-Marin, E. S. Martinez-Gomez, E. S. Hernandez-Gress, N. Hernandez-Romero, and V. Volpi-Leon, "Cellular particle swarm optimization with a simple adaptive local search strategy for the permutation flow shop scheduling problem," *Arch. Control Sci.*, vol. 29, no. 2, pp. 205–226, 2019.
[33] E. S. Hernández-Gress, J. C. Seck-Tuoh-Mora, N. Hernández-Romero, J. Medina-Marín, P. Lagos-Eulogio, and J. Ortíz-Perea, "The solution of the concurrent layout scheduling problem in the job-shop environment through a local neighborhood search algorithm," *Expert Syst. Appl.*, vol. 144, Apr. 2020, Art. no. 113096.
[34] N. J. Escamilla Serna, J. C. Seck-Tuoh-Mora, J. Medina-Marin, N. Hernandez-Romero, I. Barragan-Vite, and J. R. Corona Armenta, "A global-local neighborhood search algorithm and Tabu search for flexible job shop scheduling problem," *PeerJ Comput. Sci.*, vol. 7, p. e574, May 2021.
[35] Y. Shi, H. Liu, L. Gao, and G. Zhang, "Cellular particle swarm optimization," *Inf. Sci.*, vol. 181, no. 20, pp. 4460–4493, 2011.
[36] P. Lagos-Eulogio, J. C. Seck-Tuoh-Mora, N. Hernandez-Romero, and J. Medina-Marin, "A new design method for adaptive IIR system identification using hybrid CPSO and DE," *Nonlinear Dyn.*, vol. 88, no. 4, pp. 2371–2389, Jun. 2017.
[37] G.-G. Wang, A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "A novel improved accelerated particle swarm optimization algorithm for global numerical optimization," *Eng. Comput.*, vol. 31, no. 7, pp. 1198–1220, 2014.
[38] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-Qaness, and A. H. Gandomi, "Aquila optimizer: A novel meta-heuristic optimization algorithm," *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107250.
[39] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, "Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems," *Appl. Intell.*, vol. 51, pp. 1531–1551, Sep. 2020.
[40] Q. Askari, M. Saeed, and I. Younas, "Heap-based optimizer inspired by corporate rank hierarchy for global optimization," *Expert Syst. Appl.*, vol. 161, Dec. 2020, Art. no. 113702.
[41] J. Brest, M. S. Maucec, and B. Boskovic, "IL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1188–1195.
[42] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 145–152.
[43] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine predators algorithm: A nature-inspired Metaheuristic," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113377.
[44] Q. Askari, I. Younas, and M. Saeed, "Political optimizer: A novel socio-inspired meta-heuristic for global optimization," *Knowl.-Based Syst.*, vol. 195, May 2020, Art. no. 105709.
[45] V. Plevris, N. P. Bakas, and G. Solorzano, "Pure random orthogonal search (PROS): A plain and elegant parameterless algorithm for global optimization," *Appl. Sci.*, vol. 11, no. 11, p. 5053, May 2021.
[46] M. Shaqfa and K. Beyer, "Pareto-like sequential sampling heuristic for global optimisation," *Soft Comput.*, vol. 25, no. 14, pp. 9077–9096, Jul. 2021.
[47] A. Baykasoğlu and A. Kpinar, "Weighted superposition attraction (WSA): A swarm intelligence algorithm for optimization problems—Part 1: Unconstrained optimization," *Appl. Soft Comput.*, vol. 56, pp. 520–540, Jul. 2017.
[48] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Comput. Methods Appl. Mech. Eng.*, vol. 191, pp. 1245–1287, Jan. 2002.
[49] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm Evol. Comput.*, vol. 56, Aug. 2020, Art. no. 100693.
[50] E. Zahara and Y.-T. Kao, "Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3880–3886, Mar. 2009.
[51] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Inf. Sci.*, vol. 178, no. 15, pp. 3043–3074, Aug. 2008.
[52] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan, "Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique," *Struct. Multidisciplinary Optim.*, vol. 37, no. 4, pp. 395–413, 2009.

[53] L. Wang and L.-P. Li, "An effective differential evolution with level comparison for constrained engineering design," *Struct. Multidisciplinary Optim.*, vol. 41, no. 6, pp. 947–963, Jun. 2010.
[54] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, "Water cycle algorithm—A novel metaheuristic optimization method for solving constrained engineering optimization problems," *Comput. Struct.*, vols. 110–111, pp. 151–166, Nov. 2012.
[55] F. Hamza, H. Abderazek, S. Lakhdar, D. Ferhat, and A. R. Yildiz, "Optimum design of cam-roller follower mechanism using a new evolutionary algorithm," *Int. J. Adv. Manuf. Technol.*, vol. 99, nos. 5–8, pp. 1267–1282, 2018.
[56] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303–315, 2011.
[57] P. Savsani and V. Savsani, "Passing vehicle search (PVS): A novel metaheuristic algorithm," *Appl. Math. Model.*, vol. 40, nos. 5–6, pp. 3951–3978, 2016.
[58] O. Sigmund, "A 99 line topology optimization code written in MATLAB," *Struct. Multidisciplinary Optim.*, vol. 21, no. 2, pp. 120–127, Apr. 2001.

JUAN CARLOS SECK-TUOH-MORA received the M.S. and Ph.D. degrees in computer science from the Center for Research and Advanced Studies, National Polytechnic Institute, Mexico, in 1999 and 2002, respectively. He is currently a Professor-Researcher of the academic area of engineering with the Autonomous University of Hidalgo, Hidalgo, Mexico. He is also the National Researcher Level 2 within the National System of Researchers of CONACYT. His current research interests include cellular automata, metaheuristics, evolutionary algorithms, neural networks to model, design, optimize, and control engineering systems.

OMAR LOPEZ-ARIAS received the B.S. degree in electrical engineering from the Technological Institute of Pachuca, Hidalgo, Mexico, in 2016, and the M.S. degree in automation and control sciences from the Autonomous University of Hidalgo, Hidalgo, in 2019, focused on adaptive control, where he is currently pursuing the Ph.D. degree in system modeling and analysis, developing evolutionary algorithms inspired by cellular automata rules to optimize and control engineering systems.

NORBERTO HERNANDEZ-ROMERO received the M.S. degree from the Department of Electrical Engineering, Laguna Technological Institute, Mexico, in 2001, and the Ph.D. degree from the Autonomous University of Hidalgo, Hidalgo, Mexico, in 2009. He is currently a Professor-Researcher of the academic area of engineering with the Autonomous University of Hidalgo. He is also the National Researcher Level 1, within the National System of Researchers of CONACYT. His current research interests include system identification, feedback control design, fuzzy logic, neural networks, and metaheuristics algorithms and its applications.

GENARO J. MARTÍNEZ received the M.S. and Ph.D. degrees in computer science from the Center for Research and Advanced Studies, National Polytechnic Institute, Mexico City, Mexico, in 2000 and 2006, respectively. He is currently a Professor with the School of Computer Sciences, National Polytechnic Institute, and a Visiting Fellow with the Unconventional Computing Centre, University of the West of England, Bristol, U.K. His current research interests include cellular automata, unconventional computing, artificial life, complex systems, and swarm robotics.

VALERIA VOLPI-LEON received the B.S. degree in architecture from the Autonomous University of Hidalgo, Hidalgo, Mexico, in 2010, and the M.S. degree in engineering in construction from the Autonomous University of Puebla, Puebla, Mexico, in 2017. She is currently a Professor-Researcher of the academic area of engineering with the Autonomous University of Hidalgo. She has the PRODEP-SEP Desirable Profile Distinction. Her current research interests include sustainable habitability and engineering problems in construction.