Expert Systems With Applications 177 (2021) 114930

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A continuous-state cellular automata algorithm for global optimization

Juan Carlos Seck-Tuoh-Mora *, Norberto Hernandez-Romero, Pedro Lagos-Eulogio, Joselito Medina-Marin, Nadia Samantha Zuñiga-Peña

Área Académica de Ingeniería, Instituto de Ciencias Básicas e Ingeniería, Universidad Autónoma del Estado de Hidalgo, Carr Pachuca-Tulancingo Km 4.5. Pachuca, 42184 Hidalgo, Mexico

ARTICLE INFO

Keywords:
Global optimization
Cellular automata
Metaheuristics
Engineering applications

ABSTRACT

Cellular automata are capable of developing complex behaviors based on simple local interactions between their elements. Some of these characteristics have been used to propose and improve metaheuristics for global optimization; however, the properties offered by the evolution rules in cellular automata have not yet been used directly in optimization tasks. Inspired by the complexity that various evolution rules of cellular automata can offer, the continuous-state cellular automata algorithm is proposed. In this way, the algorithm takes advantage of different evolution rules to maintain a balance that maximizes the exploration and exploitation properties in each iteration. The efficiency of the algorithm is proven with 48 test problems widely used in the literature, 4 engineering applications that were also used in recent literature, and the design of adaptive infinite-impulse response filters, with the reference functions of 10 full-order filters being tested. The numerical results prove its competitiveness in comparison with state-of-the-art algorithms. The source codes of the proposed algorithm are publicly available at https://github.com/juanseck/CCAA.git.

1. Introduction

Nowadays, engineering problems have become increasingly complex due to the changes in the numbers and relationships of the variables that define them. Under these conditions, it is very difficult, or even impossible, to apply traditional mathematical methods to solve these problems quickly and satisfactorily. Conventional methods face difficulties, such as convergence to local optima and the dependence of their correct convergence on the initial solution chosen, while attempting to solve complex problems. These methods also require that the problem being solved fulfills a series of mathematical properties such as convexity, continuity, and differentiability, which often leads to an excessive simplification of the problem (Sarker, Kamruzzaman, & Newton, 2003). In many cases, the methods based on mathematical programming such as the conjugate gradient and quasi-Newton methods are not suitable for solving multidimensional, multimodal, non-continuous, and non-differentiable problems (Jamil & Yang, 2013).

For this reason, new optimization methods have been proposed, such as metaheuristics, which are high-level algorithms capable of finding adequate solutions. Although it cannot be demonstrated that metaheuristic algorithms reach an optimal value, they can generate satisfactory results in a reasonable time (Kumar & Davim, 2020).

Metaheuristic algorithms are of great relevance because, in general, they are easy to implement and their operation is based on simple concepts that do not require information regarding the gradient of the function to be optimized. A good metaheuristic algorithm can escape from local optima in multiple dimensions and be used in a wide range of optimization, design, and parameter identification problems.

For that purpose, metaheuristic algorithms must perform in an efficient and balanced way the exploration of the solution space and the exploitation of promising areas of said space. Since these actions can conflict and leave the algorithm trapped in a local minimum, maintaining the right balance between the two processes is essential to obtain a metaheuristic algorithm that calculates good solutions, avoiding stagnation or premature convergence towards local minima.

In this work, a new metaheuristic algorithm based on the neighborhood concept and evolution rules of cellular automata is proposed for the global optimization of problems defined in multiple dimensions. As this algorithm uses continuous variables, it has been named the continuous-state automata algorithm (CCAA).

The algorithm defines a set of initial solutions, each of which generates a random neighborhood of possible new solutions to improve the

* Corresponding author.
E-mail addresses: jseck@uaeh.edu.mx (J.C. Seck-Tuoh-Mora), nhromero@uaeh.edu.mx (N. Hernandez-Romero), plagos@uaeh.edu.mx (P. Lagos-Eulogio), jmedina@uaeh.edu.mx (J. Medina-Marin), zu450520@uaeh.edu.mx (N.S. Zuñiga-Peña).

https://doi.org/10.1016/j.eswa.2021.114930
Received 8 December 2020; Received in revised form 20 February 2021; Accepted 19 March 2021
Available online 29 March 2021

fitness (or cost) of these solutions concerning the problem to be optimized, which, in the case of this work, is the minimization of functions. This neighborhood is formed using rules inspired by cellular automata, where some of them serve to exchange information with other solutions while others only take information from the solution and the best cost obtained to induce changes in its elements.

The motivation of the CCAA is found in the operation of the swarm intelligence (SI) algorithms, where a set of individuals, particles, or agents (in our case, they are called smart-cells) interact locally, sharing information following a set of simple rules but producing a collective emergent behavior capable of optimizing complex problems (Hassanien & Emary, 2018).

The vast majority of SI algorithms are inspired by the complex behavior that emerges between the members of a natural system. These metaheuristics are simple in conceptualization and implementation, starting with a population that evolves towards an optimal or near-optimal solution. These have the advantage of not depending on the initial solution that is chosen, do not require the satisfaction of specific mathematical properties of the problem, and are flexible enough to be applied to a great variety of problems (Cui & Gao, 2012).

In our case, the algorithm presented in this research is inspired by the behavior that arises from an artificial system, particularly from cellular automata. The objective of this paper is in three aspects: (a) Provide a general presentation of a new optimization method called the CCAA, showing the functioning of a cellular automaton, and then proposing various systems' rules to obtain a general-purpose optimization algorithm; (b) Contrast the CCAA with other recent proven metaheuristics by comparing their efficiencies with a set of test functions; (c) Use the CCAA in real-world problems to demonstrate its applicability and efficiency.

The proposed algorithm's strength is that its specification is simple, which facilitates its computational programming and, therefore, its application for optimization tasks, offering satisfactory results compared to other recently published metaheuristics with performances that are known to be efficient. The contribution of the CCAA lies in showing that the simple interaction rules of cellular automata, capable of generating a great diversity of dynamic behaviors, can also be a source of inspiration for the conception of new metaheuristic algorithms suitable for optimization tasks. These rules lead to the specification of an algorithm that is very simple to implement and which, at the same time, is highly competitive for global optimization in several dimensions.

Metaheuristics have been used in other engineering applications such as the identification of system parameters from experimental data, the optimal modeling of materials, and multi-dimensional optimization, to mention a few (Kumar & Davim, 2020). Operation of the CCAA is also tested in some of these applications to check its performance against other specialized algorithms, obtaining satisfactory results. These applications are gear train design (Sandgren, 1990), pressure vessel design (Salih, Alsewari, & Yaseen, 2019), welded bean design (Coello, 2000), cantilever beam design (Mirjalili, 2015), and the design of adaptive digital filters, one of the more popular application fields in recent years due to its relevance to real-world problems (Jiang, Wang, & Ji, 2015; Lagos-Eulogio, Seck-Tuoh-Mora, Hernandez-Romero, & Medina-Marin, 2017; Atul Kumar Dwivedi & Londhe, 2018).

The structure of the paper is as follows. Section 2 presents a review of the literature on the latest SI metaheuristics proposed for the global optimization of functions. Section 3 describes the basic concepts of cellular automata, the proposals for metaheuristic algorithms based on cellular automata, and the opportunity to improve this type of implementations. Section 4 presents the CCAA and explains the adaptations of various evolution rules to carry out exploration and exploitation tasks concurrently as well as the general strategy of the CCAA. Section 5 shows the experimental results of 48 test functions in 30 and 500 dimensions, presenting a statistical comparison with other 14 state-of-the-art algorithms. Section 5.4 applies the CCAA in 4 engineering problems that are commonly considered in the specialized literature, in addition to utilizing the proposed algorithm in the design of adaptive infinite-impulse response (IIR) filters of full order. These results are compared with previously published findings to show the effectiveness and performance of the CCAA. The last section provides the conclusions and future proposals for the development of the CCAA.

2. An overview of swarm intelligence metaheuristics

The optimization problem, which consists of finding the best set of values that optimize a given objective function, subject to restrictions, has been approached from different perspectives. One of them is represented by metaheuristic algorithms inspired by physical and natural systems, with variants and modifications that intensify their exploitation and exploration characteristics.

Due to the "no free lunch" (NFL) theorem (Wolpert & Macready, 1997), all the proposed optimization algorithms have an equivalent average performance if applied to all possible optimization problems. Thus, a general-purpose algorithm cannot be considered to be better than the others. This fact implies that the proposal of new metaheuristics needs to continue to have a greater possibility of options in optimizing general and particular problems. Thus, in recent years, the interest and diffusion in new, robust, easy-to-implement, and flexible algorithms has grown in order to optimize progressively complex problems with a larger number of dimensions (Dokeroglua, Ender Sevincb, & Cosarb, 2019).

The metaheuristics based on SI refer to algorithms that mimic the collective intelligence that emerges from the global behavior of systems made up of multiple interacting agents who follow some simple rules without the existence of some central control. The SI algorithms are characterized by their simplicity of implementation in a computer and their ability to use information from the swarm to solve a complex problem. Particle swarm optimization (PSO) (Eberhart & Kennedy, 1995) and ant colony optimization (ACO) (Dorigo & Gambardella, 1997; Dorigo, Birattari, & Stutzle, 2006) are two of the most emblematic SI methods.

As examples of further SI algorithms that are well-recognized for their robustness and application flexibility, we can find the artificial bee colony (ABC) algorithm (Karaboga, 2005), krill herd (KH) algorithm (Gandomi & Alavi, 2012; Wang, Gandomi, & Alavi, 2014), the cuckoo search algorithm (CSA) (Gandomi, Yang, & Alavi, 2013), the gray wolf optimizer (GWO) algorithm (Mirjalili, Mirjalili, & Lewis, 2014), the ant lion optimizer (ALO) (Mirjalili, 2015), elephant herding optimization (EHO) (Wang, Deb, & Coelho, 2015), the whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016), drone squadron optimization (DSO) (de Melo & Banzhaf, 2018), the earthworm optimization algorithm (EOA) (Wang, Deb, & Coelho, 2018), the moth search algorithm (MSA) (Wang, 2018), Harris hawks optimization (HHO) (Heidari et al., 2019), the monarch butterfly optimization (MBO) (Wang, Deb, & Cui, 2019), the mayfly optimization algorithm (MOA) (Zervoudakis & Tsafarakis, 2020), and the slime mold algorithm (SMA) (Li, Chen, Wang, Heidari, & Mirjalili, 2020). These algorithms are only a representative selection of the large number of SI algorithms that have been proposed recently, offering excellent results for the optimization of complex problems.

All metaheuristics have two stages in the process of optimizing a problem: exploration and exploitation (Salcedo-Sanz, 2016). Exploration is the phase where the metaheuristics look for solutions in the search space's entire breadth, taking advantage of its randomized operators. Exploitation is the phase where the search becomes more limited in the different areas established by the exploration phase, increasing the precision of solutions in local regions to enhance their quality. The balance of these two phases is crucial to find optimal or satisfactory solutions; this balance depends on the problem that is being solved, so a good metaheuristic must be able to adapt to the conditions of the problem where it is being applied, such as avoiding being trapped in local minima or converging premature to unsatisfactory solutions.

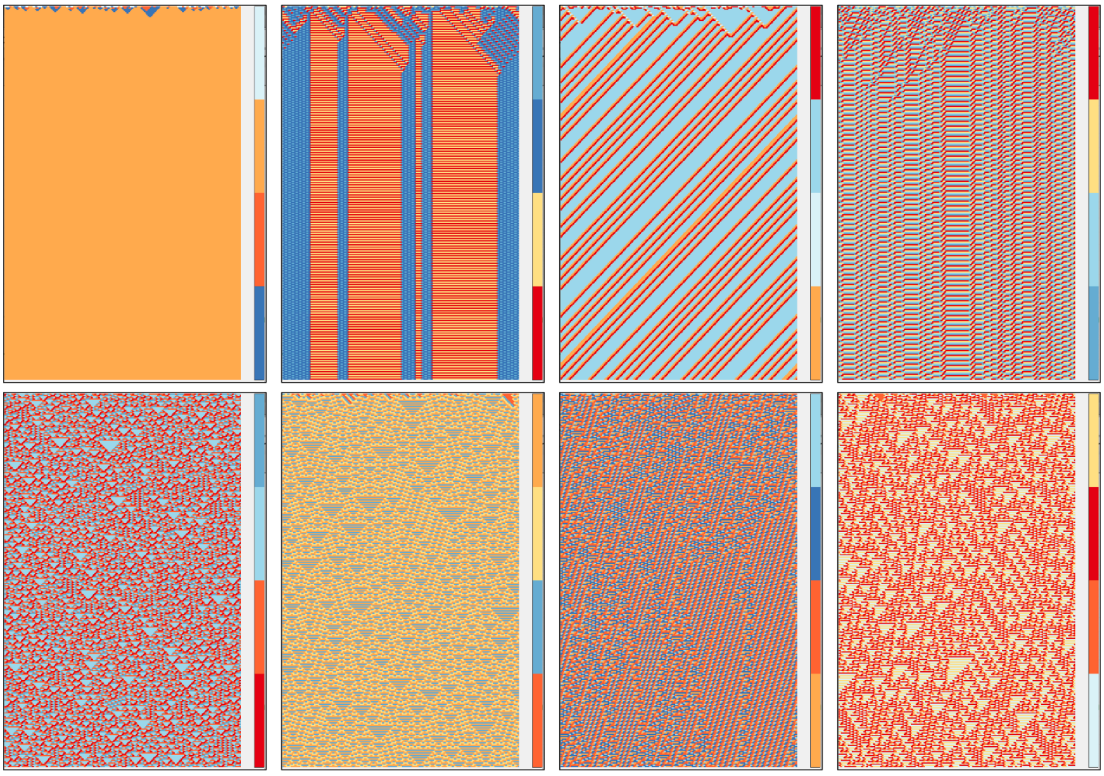

Fig. 1. Examples of different cellular automata with 4 states.

3. Basics of cellular automata

Cellular automata (CAs) are mathematical models proposed in Von Neumann and Ulam's visionary work on self-reproducing systems (von Neumann, 1966). CAs are easily implementable on a computer, and for many years, they have been investigated for their ability to produce highly diverse dynamic behaviors, in analogy to what natural systems can generate (Wolfram, 2002; McIntosh et al., 2009).

A CA is a discrete dynamical system composed of cells that initially take their values from a finite set of possible states. The dynamics of the system is given in discrete steps. In each step, a cell takes into account its current state and those of its neighbors to update its state. This assignment of neighborhoods to states is known as the evolution rule. Thus, a CA is discrete in time and space.

CAs are easy to implement in a computer, given the simplicity of their specification. However, these systems can develop the emergence of chaotic and complex global behaviors, which have been widely investigated and used to solve various computational and engineering problems (Schiff, 2011; Salcido, 2011). Fig. 1 shows the evolutions of different CAs with 4 states, with distinct evolution rules and dynamical behaviors.

Although concepts and ideas of CAs have been used before for proposing new metaheuristics, very few have used the main concept that characterize CAs, the versatility that can be found in its evolution rules. One of the most recent works that uses the concept of neighborhood in CAs to propose a global optimization metaheuristic is the cellular particle swarm optimization (CPSO) (Shi, Liu, Gao, & Zhang, 2011). CPSO has been shown to be very useful for global optimization tasks, although the proposed evolution rule continues to be based on an adaptation of the classical PSO equations. Other metaheuristics were proposed for the implementation of concepts of CAs oriented to particular engineering problems, such as task scheduling (Seredynski & Zomaya, 2002), design of adaptive IIR filters (Lagos-Eulogio et al., 2017), and the simultaneous optimization of the plant layout and the sequencing of tasks in a job-shop-type system (Hernández-Gress et al., 2020).

However, the versatility of the evolution rules that are widely used in CAs for various modeling problems and for computational complexity (Hoekstra, Kroc, & Sloot, 2010; Bilan, Bilan, & Motornyuk, 2020) has not been adapted in an algorithm for global optimization tasks. This is the main contribution of this paper, whose implementation is described in the next section.

4. Continuous-state automata algorithm (CCAA)

The essence of the CCAA is to emulate the dynamic behavior of a CA. Each solution (or smart-cell) is a cell of the CA, with its state defined by its current position's values. The smart-cells' dynamic behavior is defined by different evolution rules selected at random from a set of available rules, where each rule produces a new solution. Different evolution rules generate distinct new solutions, from which, the one with the least cost is selected to replace the current smart-cell. Thus, the entire population improves its positions as the system evolves in each iteration of the optimization process.

Each smart-cell is represented by its position or the set of values to be refined in order to minimize the objective function. This representation allows each smart-cell to evolve taking information from its

environment to amend its values in the optimization process.

There are evolution rules that need another neighbor to generate a new solution; this neighbor is another smart-cell that is chosen randomly as well. Other rules need only the best cost obtained so far to change the values of the smart-cell, while other rules require the only information of the values that the smart-cell contains.

The randomness in selecting rules allows each smart-cell to have access to the information of all the other smart-cells to generate a neighborhood of new solutions. Depending on the selected rule, there is a large or small change in the position of the smart-cell in every neighbor. This structure allows for the exploration of the search space in the initial stages of optimization and the refinement of the positions of smart-cells in the exploitation stage.

The function that is to be optimized (in this work, the case of minimization is taken) is defined as $f(\boldsymbol{x})\rightarrow\mathbb{R}$ in n dimensions, where $\boldsymbol{x}\in\mathbb{R}^n$ and each $x(i)$ value in $\boldsymbol{x}$ is bounded between a lower bound $lb(i)$ and an upper bond $ub(i)$. If the bounds are the same for all the values of $\boldsymbol{x}$, then these limit values will simply be stated as lb and ub. The CCAA first takes an initial population S of n_S initial smart-cells. Each smart-cell is represented as $\boldsymbol{s}\in\mathbb{R}^n$ and its cost is given by $f(\boldsymbol{s})$. Given a population S, the best solution will be represented as $\boldsymbol{b}_S$ such that its cost $f(\boldsymbol{b}_S)$ is minimal with respect to all the other smart-cells in S.

The algorithm uses a total of 10 evolution rules to be applied on the smart-cells in S and their costs in f. The general strategy is to generate a neighborhood of each smart-cell with m neighbors (or new solutions) and from that neighborhood, take the best solution as the new smart-cell in a probabilistic way or if it improves the cost of the original smart-cell. The different rules' parameters are focused so that a smart-cell can move away or make small changes from its current position or move closer to or further away from another smart-cell. Thus, the criterion for selecting the most appropriate CCAA parameters is to balance the changes generated by the rules on smart-cells to obtain a more efficient optimization process. The evolution rules used to carry out the CCAA's exploration and exploitation actions are described in the upcoming section.

4.1. Rules for approaching a neighbor

In this type of rule, a smart-cell s_i will approach another smart-cell s_j depending on the cost $f(s_i)$ and $f(s_j)$. The structure of the rule is as follows:

Algorithm 1: Approach rule

Result: New smart-cell $\boldsymbol{evol}$
Input: $\boldsymbol{s}_i$, $\boldsymbol{s}_j$, $f(\boldsymbol{s}_i)$, $f(\boldsymbol{s}_j)$, $prop$;
$\boldsymbol{evol} = \boldsymbol{s}_i$;
if $condition(f(\boldsymbol{s}_i), f(\boldsymbol{s}_j))$ **then**
  $\boldsymbol{dec} = (\boldsymbol{s}_i - \boldsymbol{s}_j) * prop * rand$;
end
$\boldsymbol{evol} = \boldsymbol{evol} - \boldsymbol{dec}$;

The rule in Algorithm 1 is very simple: if $condition(f(s_i), f(s_j))$ is true, a vector is calculated with the difference between each element of s_i and s_j and the rule weights these differences by a proportion that goes between 0 and $prop$ in a uniform, random way. If the condition is false, s_i remains unchanged.

This makes s_i get closer to s_j as $prop$ indicates; if this value is low, the approach will be small. This rule serves the purposes of differentiated exploitation of the elements in s_i, since the vector of differences is calculated element by element. In the CCAA, one version of this rule, R_1, is used, which takes $f(s_i) \neq f(s_j)$ as condition, and $prop = lower_p$, where $lower_p$ has a low value. This allows the original s_i to be moved in its close neighborhood in the direction of a neighbor with a different cost.

4.2. Rules for staying away from a neighbor

Algorithm 2: Recede rule

Result: New smart-cell $\boldsymbol{evol}$
Input: $\boldsymbol{s}_i$, $\boldsymbol{s}_j$, $f(\boldsymbol{s}_i)$, $f(\boldsymbol{s}_j)$, $prop$;
$\boldsymbol{evol} = \boldsymbol{s}_i$;
if $condition(f(\boldsymbol{s}_i), f(\boldsymbol{s}_j))$ **then**
  $\boldsymbol{inc} = (\boldsymbol{s}_i - \boldsymbol{s}_j) * prop * rand$;
end
$\boldsymbol{evol} = \boldsymbol{evol} + \boldsymbol{inc}$;

The rule in Algorithm 2 is also simple. If $condition(f(s_i), f(s_j))$ is true, the vector of differences $s_i - s_j$ is calculated and weighted with a uniform random value between 0 and $prop$. If the condition is false, s_i does not change.

This rule has the effect of moving s_i away from s_j, which implies that if s_i is close to s_j, then the rule can continue to exploit areas further away from s_j and help to escape local minima. In the other case, where s_i is far from s_j, then the new solution will be far from s_j and will serve the exploration of new areas in the search space for both s_i and s_j.

In the CCAA, two versions of this rule are used: R_2 with the condition $f(s_i) \neq f(s_j)$ and $prop = upper_p$ is considered, where $upper_p$ has a high value that allows for a significant difference from s_j, and R_3, which takes as condition $f(s_i) < f(s_j)$ and $prop = lower_p$, causing s_i to deviate a little from s_j only when the cost $f(s_j)$ is higher. This rule serves to intensify the task of exploiting the information in s_i in a direction away from s_j.

4.3. Rules for changes in a smart-cell

Algorithm 3: Change rule

Result: New smart-cell $\boldsymbol{evol}$
Input: $\boldsymbol{s}_i$, $\boldsymbol{s}_j$, $f(\boldsymbol{s}_i)$, $f(\boldsymbol{s}_j)$, $dist$;
$\boldsymbol{evol} = \boldsymbol{s}_i$;
$sum = f(\boldsymbol{s}_i) + f(\boldsymbol{s}_j)$;
$pond = 1 - (f(\boldsymbol{s}_j)/sum)$;
$r = (rand * dist) - (dist/2)$;
forall k *in* $length(\boldsymbol{evol})$ **do**
  if $rand <= pond$ **then**
    $\boldsymbol{evol}(k) = \boldsymbol{evol}(k) + r * \boldsymbol{s}_j(k)$;
  end
end

The rule in Algorithm 3 is to estimate how optimal s_i is compared to s_j taking into account their costs. If $f(s_i)$ is very high compared to $f(s_j)$, then *pond* will have a higher value; this will imply a greater probability that each element of s_i will be modified. This probability of change will decrease as the value of $f(s_i)$ becomes lower, which is expected in the optimization process. The change consists of increasing the element $s_i(k)$ with some influence of $s_j(k)$. This weight depends on the parameter *dist* with $k = 1,\ldots,n_d$, where n_d is the number of dimensions of the problem to be optimized.

The effect of this rule is to cause a change, element by element, in s_i, taking as a factor of change the neighbor s_j. Therefore, it is a rule focused on exploring new areas of the search space, mainly if the value of $f(s_i)$ is very high compared to $f(s_j)$.

In the CCAA, two versions of this rule are used: R_4, where the parameter $dist = upper_d$, with $upper_d$ being a value that allows extensive changes, and R_5, with $dist = lower_d$ and $lower_d$, defined as a smaller value that allows moderate changes.

4.4. Rules for increasing values in a smart-cell

Algorithm 4: Increment rule

Result: New smart-cell $\boldsymbol{evol}$
Input: $\boldsymbol{s}_i$, $f(\boldsymbol{s}_i)$, $f(\boldsymbol{b}_S)$, $dist$;
$\boldsymbol{evol} = \boldsymbol{s}_i$;
$sum = f(\boldsymbol{s}_i) + f(\boldsymbol{b}_S)$;
$pond = 1 - (f(\boldsymbol{s}_i)/sum)$;
$r = (rand * dist) - (dist/2)$;
forall k *in* $length(\boldsymbol{evol})$ **do**
  if $rand <= pond$ **then**
    $\boldsymbol{evol}(k) = (1 + r) * \boldsymbol{evol}(k)$;
  end
end

The rule in Algorithm 4 is very similar to the change rule, but in its definition, it only takes into account the best cost obtained in the population $f(\boldsymbol{b}_S)$ and an increment parameter *dist*. The weighting is done taking into account $f(\boldsymbol{s}_i)$ and $f(\boldsymbol{b}_S)$. If $f(\boldsymbol{s}_i)$ is larger, then *pond* will be small, inducing little change in $\boldsymbol{s}_i$. If $f(\boldsymbol{s}_i)$ is close to $f(\boldsymbol{b}_S)$, the probability of making changes to the elements of $\boldsymbol{s}_i$ will be increased. The applied changes will also be in proportion to the same values of $\boldsymbol{s}_i$ weighted by the *dist* parameter. This rule's effect is to induce self-generated changes by the values $\boldsymbol{s}_i$, so it is a rule focused on exploiting the information of $\boldsymbol{s}_i$, especially when it has a better cost. In the CCAA, two versions of this rule are used: R_6, which applies a parameter $dist = upper_d$ to induce larger increments on the values of $\boldsymbol{s}_i$, and R_7, which uses $dist = lower_d$ to only cause small modifications to the smart-cell.

4.5. Rules for majority values in a smart-cell

Algorithm 5: Majority rule

Result: New smart-cell $\boldsymbol{evol}$
Input: $\boldsymbol{s}_i$, $dist$;
$elem =$ most repeated element in $\boldsymbol{s}_i$;
forall k *in* $length(\boldsymbol{s}_i)$ **do**
  $\boldsymbol{cam}(k) = \boldsymbol{s}_i(k) - elem$;
end
$\boldsymbol{cam} = \boldsymbol{cam} * dist * rand$;
$\boldsymbol{evol} = \boldsymbol{s}_i - \boldsymbol{cam}$;

The rule in Algorithm 5 is very simple. It takes the element *elem* that is most repeated in $\boldsymbol{s}_i$ and forms a vector $\boldsymbol{cam}$ of differences between $\boldsymbol{s}_i$ and *elem* weighted by a parameter between 0 and *dist*. The vector $\boldsymbol{cam}$ is subtracted from $\boldsymbol{s}_i$ to form a new solution in order to bring the values of $\boldsymbol{s}_i$ closer to the most repeated value.

This rule's effect is to autogenerate a change in $\boldsymbol{s}_i$ that makes its values more homogeneous, which is suitable for many optimization problems where vectors with good costs have elements with the same value.

In the CCAA, two versions of this rule are used: R_8, which takes the most repeated element of $\boldsymbol{s}_i$ and the analogous R_9, which can be considered as a minority rule as it chooses the least repeated element of $\boldsymbol{s}_i$.

4.6. Rules for rounding values in a smart-cell

Algorithm 6: Rounding rule

Result: New smart-cell $\boldsymbol{evol}$
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 k *in* $length(\boldsymbol{evol})$ **do**
  if $rand <= pond$ **then**
    $\boldsymbol{evol}(k) = round(\boldsymbol{evol}(k), n_r)$;
  end
end

The rule in Algorithm 6 takes the weight of $f(s_i)$ in contrast to that of $f(\boldsymbol{b}_S)$. If $f(s_i)$ is large, then *pond* is small, and few changes are made; otherwise, there is a greater probability that changes will occur. Each change consists of rounding selected elements of $\boldsymbol{s}_i$ to discretize the decimal part to as many digits as indicated by n_r.

This rule's effect is to autogenerate a change in $\boldsymbol{s}_i$ that rounds its values, which is useful for optimization problems being primarily

focused on finding values for a system's parameter specifications. In the CCAA, a single version R_{10} of this rule is used. Given the previous rules, Algorithm 7 presents the general structure of the CCAA.

4.7. Complete structure of the CCAA

Algorithm 7: The continuous-state cellular automata algorithm (CCAA)

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 $lower_p$, $upper_p$, $lower_d$, $upper_d$, $lower_r$ y $upper_r$;
Generate random population S of n_S smart-cells;
Evaluate S in f;
forall $i = 2$ *to* n_{it} **do**
  Keep the best n_{el} smart-cells in a new population;
  forall $j = n_{el} + 1$ *to* n_S **do**
    Take $\boldsymbol{s}_j$ and another random $\boldsymbol{s}_r$ from S for rules requiring an extra smart-cell;
    forall $k = 1$ *to* n_{ne} **do**
      Choose a rule R in a random manner;
      Obtain $\boldsymbol{evol}_k = R(\boldsymbol{s}_j, \text{additional parameters of the rule})$;
      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 $\boldsymbol{evol}$ from the k generated neighbors having a minimum cost $f(\boldsymbol{evol})$;
    if $rand < 0.5$ *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 $f(\boldsymbol{b}_S)$;

The CCAA receives 4 input values to operate, namely, the number n_S of smart-cells, the number n_{ne} of neighbors of each smart-cell, the number of iterations n_{it}, and the number of elitist solutions n_{el}. The other input values correspond to properties of the function that is to be optimized, such as the limits $lb$and ub for every element of each smart-cell and the number n_d of dimensions or values of each smart-cell.

The CCAA has 6 parameters, $lower_p$ and $upper_p$ to define the *prop*-parameter in the rules based on Algorithms 1 and 2; the $lower_d$ and $upper_d$ parameters to define the *dist* parameter for the rules arising from Algorithms 3, 4, and 5; and finally, the parameters $lower_r$ and $upper_r$ to generate a random number between these two values that defines the parameter n_r for the rule specified by Algorithm 6.

The CCAA's strategy consists of generating a random population of smart-cells, qualifying it, and taking some elitist solutions to update the population. The rest of the solutions are updated one by one using a neighborhood of possible new smart-cells, taking the available rules at random. A graphical description of how rules can update the position of a smart-cell is shown in Fig. 2. In part (*A*), the rules are applied randomly to generate new smart-cells that are possibly near or far from the original smart-cell. The neighbors are produced either by exchanging information with other smart-cells, taking as a reference the best fitness value obtained, or by taking the information of the smart-cell.

Once the neighborhood has been generated, the best position with minimum cost is selected (Fig. 2 (*B*)) to upgrade the smart-cell if it improves its cost, or otherwise with a probability of 50%. The elitism of the CCAA serves to preserve the information of the best solutions generated by the optimization process. The probability of substituting a smart-cell for another with a worse value also helps the algorithm avoid stagnation at local minima. Fig. 3 shows the CCAA flow chart.

5. Experimental results

Table 1 lists the 48 test functions used to validate the effectiveness of CCAA.

The first 14test functions are unimodal and are useful in the evaluation of the exploitation properties of the CCAA. The following 17test functions are multimodal and used to analyze the CCAA's ability to explore and escape local minima. The last 17 test functions have fixed dimensionality with a lower number of local minima and are useful in the observation of the balance between the exploration and exploitation actions of the CCAA. The definition of the test problems can be consulted in Jamil and Yang (2013) and Wang, Gandomi, Yang, and Alavi (2014), and most of their computational implementation can be found at htt

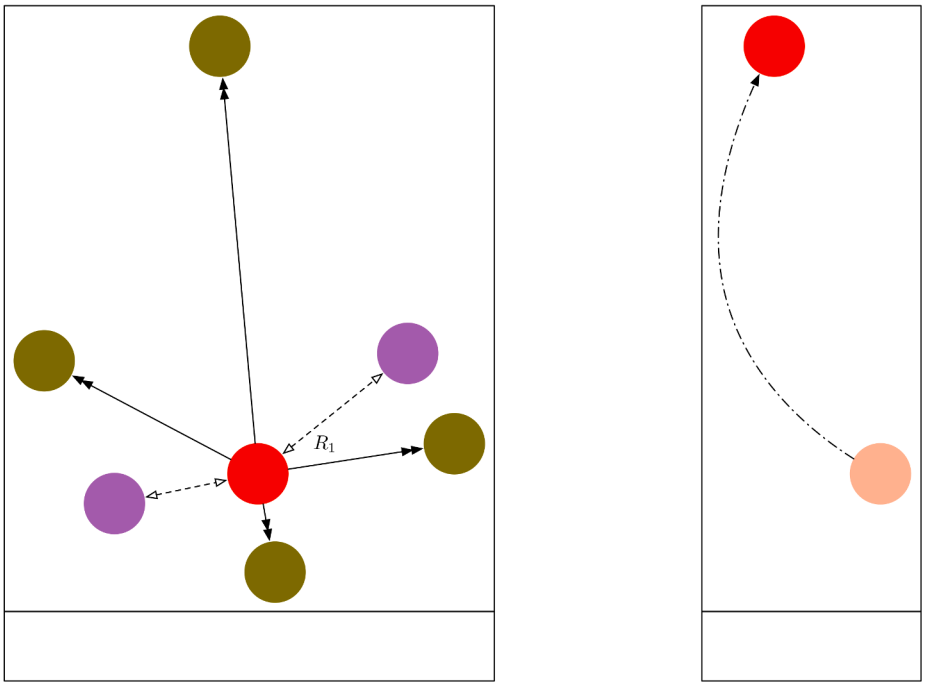

Fig. 2. Description of a neighborhood of each smart-cell in the CCAA.

p://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/go.htm, http://www.sfu.ca/ssurjano/optimization.html, http://benchmarkfcns.xyz, and http://infinity77.net/global_optimization/test_functions.html#test-functions-index.

The CCAA was implemented in Matlab 2015a, on a machine with Intel Xeon CPU at 3.1 GHz, 64 GB in the RAM, and 1 TB in the hard disk, running on the Mac OS X Catalina operating system. The parameters defining the population size and number of iterations of the CCAA (n_S, n_{ne} and n_{it}) were taken to have parameters similar to other previous studies used for comparison. For the additional parameters of the CCAA ($n_{el}, lower_p, upper_p, lower_d, upper_d, lower_r$ and $upper_r$), a preliminary experiment was conducted that considered two levels of each parameter and applied to the problems F_2 (unimodal), F_{23} (multimodal) in 30 dimensions, and F_{40} (fixed dimensions) to select the best combination.

For n_{el}, the values 1 and 2 were studied; for $lower_p$, 0.5 and 1were taken; for $upper_p$, 2 and 3 were tested. For $lower_d$, the values 0.15 and 0.3 were verified; for $upper_d$, 1 and 2 were analyzed. For $lower_r$, 1 and 2were taken and for $upper_r$, 3 and 4 were tested; having a total of 128 combinations. For each combination, 30 independent runs of the CCAA were made in the three functions selected, and the combination of parameters that obtained the best average value was chosen. As a result of this analysis, the parameters selected to conduct the comparison of the CCAA with the other algorithms in the following experiments are in Table 2.

The proposed algorithm uses 10 different rules selected randomly for each smart-cell in each iteration in order to obtain a neighborhood of new solutions and select the best one. To identify what rules are making the key contribution to the optimization process, we can classify and weight the rules depending on whether they are useful for exploring or exploiting the search space. The rules for exploration are 4 (R_2, R_4, R_6 and R_9) and the other 6rules are focused on exploitation tasks ($R_1, R_3, R_5, R_7, R_8, R_{10}$).

To detect what type of rules are working in the optimization process, the following experiment was performed. For a selected test function, the CCAA was run with the parameters in Table 2. In each iteration, the best neighbor is detected; if it improves the smart-cell's current cost, a record is kept that accumulates a value of 10in case an exploration rule was used, and 1 if an exploitation rule was employed. In this way, if the optimization process is adequate, the record should have large values at the beginning of the optimization process when the exploration phase is more intense, and decrease as the process advances when the exploitation becomes more relevant.

This experiment was applied to the same functions used to fine-tune the algorithm parameters (F_2, F_{23}, and F_{40}). For each test function, 30independent runs were made. The weights of the rules with the best result were averaged for each iteration. The results are presented in Fig. 4.

As shown in Fig. 4, in the initial iterations, the exploration rules have a strong influence on the optimization process. As the process progresses, exploitation rules obtain the best results. From the graph of the test function F_{23}, it can be seen that the exploration rules are dominant at the beginning and at a subsequent stage. This feature is suitable for multimodal functions in order to escape from local minima, showing the appropriate exploration ability that the CCAA has to improve solutions in the optimization process.

Two groups of experiments were performed. The first group evaluated the CCAA with the 31 scalable test problems in 30 dimensions and the 17 problems with fixed dimensions. The second group used the 31 scalable problems in 500 dimensions as well as the 17 fixed-dimensional problems.

The results are compared with 14 other recently published algorithms recognized by their robustness and efficiency, namely, the adaptive logarithmic spiral-Levy firefly algorithm (ADIFA) (Wu et al.,

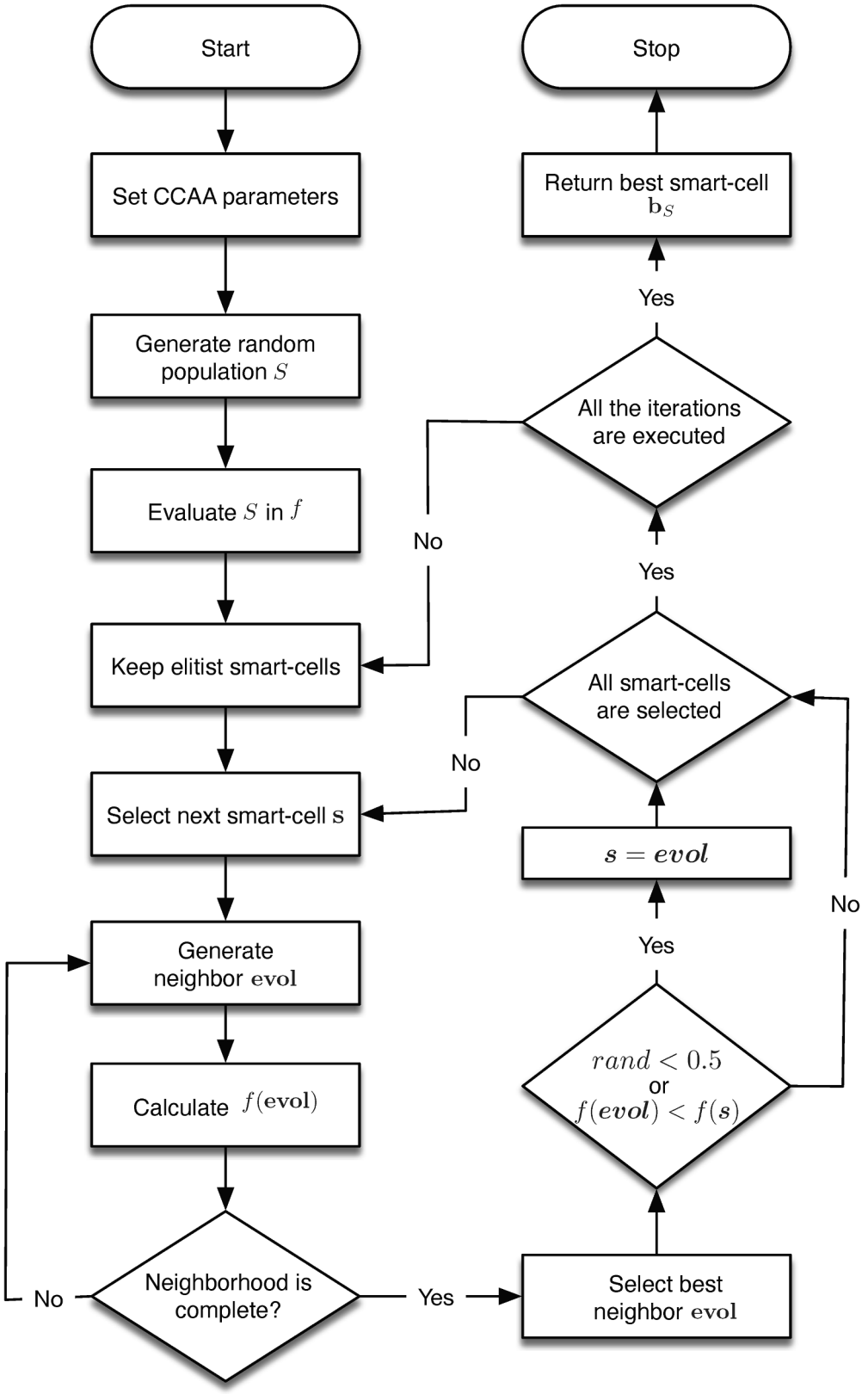

Fig. 3. CCAA flow chart.

Table 1
Benchmark functions.

No.	Name	No.	Name
F1	Brown	F25	Rastrigin
F2	Dixon & Price	F26	Schwefel 2.26
F3	Powell 2	F27	Salomon
F4	Powell Sum	F28	Schaffer 6
F5	Quartic	F29	V Sine Wave
F6	Quartic with noise	F30	XinSheYang 3
F7	Rosenbrock	F31	Wavy 1
F8	Schwefel 1.2	F32	Beale
F9	Schwefel 2.21	F33	Branin
F10	Schwefel 2.22	F34	Colville
F11	Sphere	F35	Corana
F12	Step 2	F36	Foxholes
F13	Sum Squares	F37	Goldstein-Price
F14	Zakharov	F38	Hartman 1
F15	Ackley 1	F39	Hartman 2
F16	Alpine	F40	Helical Valley
F17	Cosine Mixture	F41	Hump
F18	Griewank	F42	Kowalik
F19	Conditioned Elliptic	F43	Shekel 1
F20	Levy	F44	Shekel 2
F21	Pathological	F45	Shekel 3
F22	Penalty 1	F46	Trid 6
F23	Penalty 2	F47	Watson
F24	Periodic	F48	Wolfe

Table 2
Parameter settings of the CCAA.

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	
Lower proportion ($lower_p$)	1
Upper proportion ($upper_p$)	2
Lower distance ($lower_d$)	0.3
Upper distance ($upper_d$)	1
Lower rounding ($lower_r$)	1
Upper rounding ($upper_r$)	4

2020), the cellular particle swarm optimization (CPSO) (Shi et al., 2011), the elephant herding optimization (EHO) (Wang et al., 2015), the gray wolf optimizer (GWO) (Mirjalili et al., 2014), the Harris hawks optimization (HHO) (Heidari et al., 2019), the success-history-based adaptive differential evolution with linear population size reduction (LSHADE) (Tanabe & Fukunaga, 2014), the LSHADE with semi-parameter adaptation hybrid with a covariance matrix adaptation evolution strategy (LSPACMA) (Mohamed, Hadi, Fattouh, & Jambi, 2017), the mayfly optimization algorithm (MA) (Zervoudakis & Tsafarakis, 2020), the monarch butterfly optimization (MBO) (Wang et al., 2019), the moth search algorithm (MSA) (Wang, 2018), the modified sine cosine algorithm (MSCA) (Gupta, Deep, Mirjalili, & Kim, 2020), the double adaptive random spare reinforced whale optimization algorithm (RDWOA) (Chen, Yang, Heidari, & Zhao, 2020), the slime mould algorithm (SMA) (Li et al., 2020), and the whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016).

Most of the codes and all the parameters for these algorithms were taken from the sites noted in the references, so these are mostly implementations made by the same authors of the indicated papers, which provides a more objective comparison between algorithms. The only algorithm implemented was the MSCA, for which the original SCA code was taken and modified following the instructions published in Gupta et al. (2020). Comparison with the other algorithms shows the average value and the standard deviation of their values obtained in the objective functions, taking 30 independent runs of each algorithm.

5.1. Experiment 1: Functions on 30 dimensions and functions with fixed dimensions

This section compares the performance of the CCAA against 14 other recently published algorithms for global optimization, using 31 scalable test functions that were tested in this experiment with 30dimensions and other 17functions of fixed dimensions, which are listed in Table 1.

In this experiment, 30 independent runs were made per algorithm; for the CPSO and the CCAA, $n_S = 12$ and $n_{ne} = 6$ were used; for the rest of the algorithms, $(n_S)(n_{ne}) = 72$ individuals were used. In all algorithms, $n_{it} = 500$ was employed. Table 3 presents the results of the comparison with respect to the average and standard deviation for the unimodal functions, Table 4 shows the same values for multimodal functions, and Table 5 presents the results for functions on fixed dimensions.

For unimodal problems, the CCAA obtained 12 of the 14 best results with respect to the average value; i.e., the CCAA was able to achieve the optimal solutions for the problems $F1$ to $F5$, and from $F8$ to $F14$. Furthermore, in 11 of 14 cases, the CCAA produced the best values concerning the standard deviation, which gives evidence of the proposed algorithm's information exploitation capacity.

For the 17 multimodal problems, the CCAA also obtained 9 of the 17best average values, ($F15, F17$ to $F20, F22$ to $F23, F26$ and $F29$). In 5of them, the optimal values were calculated ($F17$ to $F19, F26$, and $F29$). The performance of the CCAA is comparable to the HHO and the SMA, and is only behind the MSCA, which produced 11 better average values. In 10 out of 17 cases, the CCAA achieved the best values referring to the standard deviation, demonstrating the CCAA's exploration ability.

For the 17 fixed-dimension problems, the CCAA produced 9 of the best average values ($F33$ to $F36, F38$ to $F40, F43$, and $F48$). Optimal value was reached in 7 of them ($F33$ to $F35, F38$ to $F40$, and $F48$). The performance of the CCAA is comparable to that of the LSHADE, which obtained 9 best average values as well, and is only behind the LSPACMA, which produced 12 better average values. In 6 out of 17 cases, the CCAA calculated the best values for the standard deviation, showing the robustness of the CCAA in carrying out exploration and exploitation actions simultaneously. Table 6 presents the results on the range that each algorithm obtained with respect to its average value in each test problem, for 30dimensions and the functions with fixed dimension. The penultimate row shows each algorithm's average rank, and the last row indicates each algorithm's final rank with regard to the average range of the 48 test problems. The CCAA's average rank is the best (2.33), followed closely by the SMA (3.15) and the MSCA (3.44).

Table 7 describes the results of the Wilcoxon rank-sum test to compare the CCAA with the other algorithms. Use of the + symbol indicates that the CCAA obtained a significantly better statistical result, a − symbol indicates a significantly worse statistical result, and the ≈symbol indicates no statistically significant difference. The last row shows the difference between the number of positive tests (+) minus the number of negative tests (−); a positive value indicates that the CCAA obtained a more significant number of positive statistical tests against the reference algorithm. In Table 7, the CCAA always achieved a favorable comparison against the other algorithms. These results show the excellent overall effectiveness of the CCAA for problems in 30dimensions and with fixed dimensions.

5.2. Experiment 2: Functions on 500dimensions and functions with fixed dimensions

The second set of experiments contemplates the same 31 scalable test problems but in 500dimensions, making the same statistical comparisons of the average and standard deviations in another 30 independent runs for each algorithm. A comparative statistical analysis was

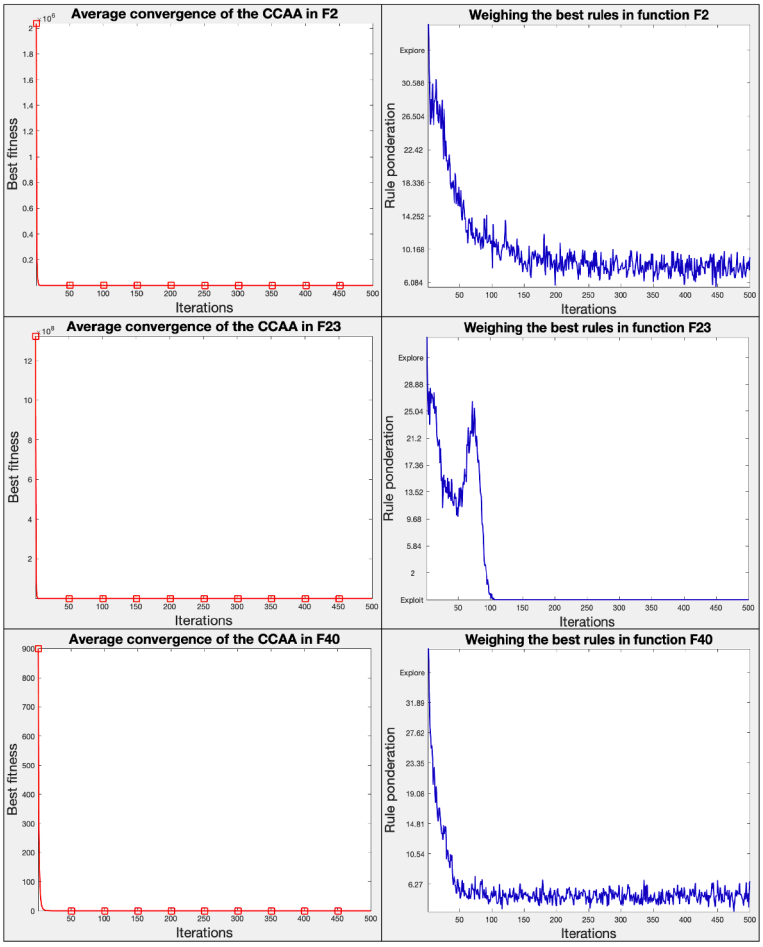

Fig. 4. Weighing the rules used in the optimization process, from exploration to exploitation.

performed in each set of experiments applying the Wilcoxon rank-sum test.

For the CPSO and the CCAA, $n_S = 12$ and $n_{ne} = 6$were used; for the rest of the algorithms, $(n_S)(n_{ne}) = 72$ individuals were applied. In all algorithms, $n_{it} = 500$ was employed. Table 8 presents the results of the comparison with respect to the mean and standard deviation for the unimodal functions, Table 9 shows the same comparison for multimodal functions, and Table 10 describes the results for functions with fixed dimensions.

For unimodal problems, the CCAA obtains 10 of the 14best average values. The CCAA was able to obtain optimal solutions for $F1, F3$ to $F5$, $F7$, and from $F10$ to $F13$. The performance of the CCAA is only behind the MSCA, which produced 11 better average values. Concerning the standard deviation, in 9 out of 14 cases, the CCAA achieved the best values, which again gives evidence of its exploitation capacity.

For the 17multimodal problems, the CCAA also produced 10 of the 17best average values. In 6of them ($F17$ to $F19, F25, F26$, and $F29$), the optimal values were calculated. The performance of the CCAA is comparable with that of the HHO and the SMA, which obtained 7 better average values, and is only behind the MSCA, which produced again 11 better average values. In 11out of 17cases, the CCAA achieved the best values referring to the standard deviation, which also shows the CCAA's exploration ability.

For the second run with 17 fixed-dimension test functions, the CCAA again produced 9 of the best average values. Optimal value was reached in 7of them ($F32$ to $F35, F39, F40$, and $F48$). The performance of the CCAA is comparable to that of the LSHADE, which produced 9 better average values, only behind the LSPACMA, which produced 10 better average values. For this experiment, the CCAA reached the lowest value in 7 of 17 cases for the standard deviation, showing once again the robustness of the CCAA to carry out exploration and exploitation actions in a balanced way.

Table 11 exposes the results on the range that each algorithm obtained with respect to its average value in each test problem, for 500 dimensions and the functions with fixed dimension. The CCAA's average rank is the best (2.08), followed by the MSCA (3.65) and the SMA (4.08).

Table 12 presents the results of the Wilcoxon rank-sum test to compare the CCAA with the other algorithms for each test function. In Table 12, the CCAA always has a favorable comparison versus the other algorithms concerning the Wilcoxon rank-sum test. These results show how effective the CCAA is in general for problems with 500 and fixed dimensions. Fig. 5 presents a sample of the convergence curves of the algorithms used in this experiment for different test problems in $30, 500$ and fixed dimensions. In these examples, only the convergence of the best algorithms is depicted in every case.

Table 3
Performance of metaheuristic algorithms compared with CCAA on 30-dimensional unimodal problems.

Benchmark		ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F1	AVG	4.22e+00	0.00e+00	2.64e-08	1.75e-09	5.84e-40	1.34e-106	3.52e-15	1.26e-29	6.63e-10	1.80e+03	1.94e-14	0.00e+00	2.39e-143	0.00e+00	3.32e-94
	STD	2.78e+00	0.00e+00	1.82e-07	4.47e-11	1.58e-39	6.26e-106	5.16e-15	1.41e-29	8.19e-10	7.20e+03	3.23e-14	0.00e+00	1.65e-142	0.00e+00	1.85e-93
F2	AVG	2.59e+02	4.51e-02	2.37e+00	9.85e-01	6.67e-01	2.49e-01	6.67e-01	6.67e-01	1.16e+00	8.15e+05	6.67e-01	4.00e-01	6.67e-01	2.02e-01	6.67e-01
	STD	2.17e+02	1.59e-01	1.17e+01	9.10e-03	5.90e-06	2.89e-03	2.35e-09	4.19e-16	1.07e+00	7.21e+05	1.41e-07	2.02e-01	6.30e-14	9.37e-02	7.92e-05
F3	AVG	1.04e+01	0.00e+00	6.38e+01	4.28e-09	1.14e-05	8.91e-102	3.56e-05	3.04e-06	6.73e-04	4.36e+03	9.23e-14	0.00e+00	8.31e-40	0.00e+00	9.73e-07
	STD	7.69e+00	0.00e+00	5.87e+01	6.83e-10	8.94e-06	6.23e-101	3.23e-05	3.28e-06	5.11e-04	3.82e+03	2.55e-13	0.00e+00	5.87e-39	0.00e+00	2.95e-06
F4	AVG	2.65e-07	0.00e+00	1.22e-24	3.99e-13	3.11e-129	2.40e-128	2.18e-37	8.81e-30	1.34e-29	1.49e+00	3.26e-21	0.00e+00	0.00e+00	0.00e+00	1.48e-133
	STD	2.86e-07	0.00e+00	5.50e-24	1.91e-13	1.71e-128	1.69e-127	1.29e-36	5.38e-29	5.18e-29	2.25e+00	4.92e-21	0.00e+00	0.00e+00	0.00e+00	1.05e-132
F5	AVG	3.78e-03	0.00e+00	2.03e-15	1.26e-21	1.64e-67	1.07e-210	4.10e-25	7.13e-56	3.26e-15	1.14e+02	9.01e-27	0.00e+00	1.88e-235	0.00e+00	5.58e-154
	STD	2.91e-03	0.00e+00	9.59e-15	1.52e-22	4.65e-67	0.00e+00	2.39e-24	1.88e-55	5.01e-15	8.31e+01	2.69e-26	0.00e+00	0.00e+00	0.00e+00	3.84e-153
F6	AVG	3.63e-01	2.25e-04	1.12e-01	3.18e-02	8.35e-04	7.30e-05	8.15e-03	6.29e-03	1.66e-02	1.27e+02	3.50e-05	4.58e-05	5.04e-05	8.67e-05	2.49e-03
	STD	1.26e-01	1.62e-04	3.77e-01	2.27e-02	4.19e-04	6.22e-05	3.13e-03	2.48e-03	5.70e-03	8.84e+01	9.41e-06	4.26e-05	4.45e-05	5.45e-05	2.66e-03
F7	AVG	7.53e+04	1.56e+00	2.07e+02	2.89e+01	2.67e+01	1.97e-03	2.14e+01	2.14e+01	5.56e+01	7.06e+07	2.66e+01	1.29e+01	2.29e+01	5.68e-01	2.71e+01
	STD	5.32e+04	6.23e+00	6.04e+02	3.23e-02	6.21e-01	2.70e-03	8.17e+00	1.06e+01	3.25e+01	8.58e+07	1.40e-01	1.35e+01	1.59e+00	5.10e-01	2.01e-01
F8	AVG	1.93e+03	0.00e+00	5.50e+03	6.93e-09	2.58e-09	1.78e-87	2.78e-01	1.29e-02	4.79e+03	6.05e+04	5.60e-14	0.00e+00	3.17e-64	1.04e-279	2.22e+04
	STD	7.81e+02	0.00e+00	4.29e+03	3.03e-09	8.64e-09	1.24e-86	2.00e-01	2.66e-02	1.98e+03	3.65e+04	1.37e-13	0.00e+00	1.46e-64	0.00e+00	8.11e+03
F9	AVG	2.57e+01	0.00e+00	1.02e+01	2.22e-05	1.55e-09	3.54e-52	1.12e-01	5.96e-02	4.67e+01	3.03e+01	5.69e-08	0.00e+00	3.87e-39	8.89e-180	3.91e+01
	STD	5.97e+00	0.00e+00	5.93e+00	4.11e-06	1.22e-09	1.83e-51	7.52e-02	3.81e-02	8.48e+00	2.28e+01	4.98e-08	0.00e+00	1.39e-38	0.00e+00	3.01e+01
F10	AVG	1.09e+14	0.00e+00	3.82e+02	2.32e-04	8.52e-21	1.40e-52	2.09e-03	2.86e-02	5.60e-03	3.54e+38	1.81e-07	0.00e+00	2.68e-87	3.67e-151	3.61e-54
	STD	4.13e+14	0.00e+00	2.26e+02	2.58e-05	8.80e-21	8.12e-52	5.42e-03	7.18e-02	2.81e-02	1.92e+39	7.53e-08	0.00e+00	9.62e-87	2.56e-150	1.19e-53
F11	AVG	1.53e+03	0.00e+00	2.78e-06	2.78e-09	3.70e-37	1.47e-100	1.47e-12	1.29e-18	6.47e-07	2.60e+04	6.18e-15	0.00e+00	2.97e-141	0.00e+00	7.49e-91
	STD	4.85e+02	0.00e+00	1.40e-05	6.64e-10	1.14e-36	1.04e-99	2.58e-12	4.32e-18	1.25e-06	2.61e+04	2.16e-14	0.00e+00	1.34e-140	0.00e+00	2.30e-90
F12	AVG	6.94e+02	0.00e+00	1.14e+01	0.00e+00	0.00e+00	0.00e+00	5.20e-01	6.60e-01	4.30e+00	2.31e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	2.23e+02	0.00e+00	1.25e+01	0.00e+00	0.00e+00	0.00e+00	7.62e-01	7.98e-01	4.53e+00	2.52e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
F13	AVG	1.86e+04	0.00e+00	6.00e+02	4.32e-08	3.77e-36	6.88e-103	8.59e-12	5.03e-17	8.13e-06	4.31e+05	1.19e-13	0.00e+00	2.24e-139	4.70e-267	2.13e-86
	STD	6.13e+03	0.00e+00	2.40e+03	9.09e-09	9.57e-36	4.75e-102	1.65e-11	1.39e-16	1.50e-05	3.40e+05	2.18e-13	0.00e+00	1.33e-138	0.00e+00	1.05e-85
F14	AVG	9.41e+01	0.00e+00	8.61e+01	1.50e-06	6.54e-15	1.36e-69	8.85e-03	4.59e-05	4.76e+01	6.93e+02	3.13e-13	0.00e+00	1.23e-65	0.00e+00	4.95e+02
	STD	2.79e+01	0.00e+00	5.85e+01	5.75e-07	1.71e-14	8.58e-69	1.04e-02	3.25e-04	5.40e+01	3.52e+02	6.01e-13	0.00e+00	3.22e-66	0.00e+00	1.16e+02

Table 4
Performance of metaheuristic algorithms compared with the CCAA on 30-dimensional multimodal problems.

Benchmark		ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F15	AVG	8.76e+00	8.88e-16	1.97e+01	1.45e-05	3.39e-14	8.88e-16	2.36e-01	2.87e-12	1.83e+00	1.50e+01	5.95e-08	8.88e-16	4.09e-15	8.88e-16	3.66e-15
	STD	2.00e+00	0.00e+00	2.22e+00	2.03e-06	4.50e-15	0.00e+00	4.62e-01	4.41e-12	5.41e-01	6.76e+00	6.80e-08	0.00e+00	1.08e-15	0.00e+00	2.41e-15
F16	AVG	3.03e+00	9.72e-12	9.19e-02	3.57e-06	2.59e-04	7.24e-55	3.00e-06	1.01e-04	1.03e-05	1.40e+01	1.07e-08	1.19e-123	1.43e-11	1.16e-161	6.09e-01
	STD	1.32e+00	4.43e-11	6.28e-01	4.07e-07	3.80e-04	3.45e-54	1.15e-05	4.94e-04	1.97e-05	1.84e+01	1.31e-08	8.40e-123	5.65e-11	8.21e-161	4.31e+00
F17	AVG	9.89e-01	0.00e+00	8.98e-01	6.53e-10	0.00e+00	0.00e+00	2.07e-02	8.87e-03	1.09e-01	5.91e+00	1.11e-13	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	3.34e-01	0.00e+00	4.33e-01	2.61e-11	0.00e+00	0.00e+00	5.18e-02	3.55e-02	1.22e-01	5.48e+00	1.37e-13	0.00e+00	0.00e+00	0.00e+00	0.00e+00
F18	AVG	1.32e+00	0.00e+00	1.74e-02	1.41e-10	1.82e-03	0.00e+00	3.01e-03	2.41e-03	1.49e-02	6.56e+00	1.93e-16	0.00e+00	2.51e-03	0.00e+00	1.19e-02
	STD	9.27e-02	0.00e+00	2.76e-02	5.02e-11	5.40e-03	0.00e+00	6.14e-03	4.52e-03	1.78e-02	5.90e+00	3.14e-16	0.00e+00	6.45e-03	0.00e+00	4.71e-02
F19	AVG	1.36e+07	0.00e+00	1.35e+06	8.43e-05	5.08e-34	1.06e-95	2.90e-07	1.03e-03	3.21e-01	4.50e+08	1.33e-11	0.00e+00	1.85e-137	1.74e-161	3.91e-86
	STD	7.19e+06	0.00e+00	2.90e+06	4.46e-05	7.99e-34	7.44e-95	1.31e-06	2.83e-03	1.31e+00	3.85e+08	1.85e-11	0.00e+00	1.16e-136	1.23e-160	1.43e-85
F20	AVG	5.29e+00	1.50e-32	1.62e+01	2.87e+00	9.45e-01	9.14e-06	1.06e-01	3.58e-03	3.18e+00	7.99e+01	4.09e-01	1.78e-07	8.64e-01	2.42e-04	1.23e-01
	STD	2.64e+00	1.38e-47	1.01e+01	8.47e-02	1.97e-01	1.19e-05	1.64e-01	1.77e-02	3.73e+00	8.79e+01	1.25e-01	2.72e-07	2.78e-01	2.02e-04	9.42e-02
F21	AVG	2.77e+00	1.05e-07	7.38e+00	5.45e-02	1.05e+01	1.58e-05	7.46e+00	7.10e+00	4.66e+00	3.51e+00	2.72e-13	7.75e-07	1.33e+00	1.13e-04	4.16e-02
	STD	1.16e+00	2.14e-07	8.49e-01	1.03e-01	7.54e-01	5.45e-05	3.47e-01	3.39e-01	7.60e-01	2.35e+00	4.98e-13	2.36e-06	9.14e-01	1.28e-04	1.98e-01
F22	AVG	1.30e+01	1.57e-32	2.08e-02	9.54e-01	1.65e-02	1.49e-06	4.15e-03	2.07e-03	3.11e-01	8.87e+07	7.45e-03	4.21e-09	7.78e-03	4.46e-04	1.95e-03
	STD	6.67e+00	5.53e-48	1.19e-01	1.81e-01	8.42e-03	2.81e-06	2.05e-02	1.47e-02	3.04e-01	1.75e+08	3.43e-03	7.60e-09	5.24e-03	6.61e-04	2.41e-03
F23	AVG	7.76e+03	1.35e-32	3.46e-01	2.99e+00	2.66e-01	1.89e-05	3.91e-03	8.79e-04	3.61e-01	1.55e+08	4.69e-01	8.65e-08	1.32e-01	1.27e-03	4.70e-02
	STD	1.41e+04	1.11e-47	1.27e+00	2.86e-03	1.65e-01	2.59e-05	9.26e-03	3.01e-03	6.64e-01	2.84e+08	1.16e-01	1.32e-07	1.41e-01	3.11e-04	2.80e-02
F24	AVG	1.04e+00	9.20e-01	1.06e+00	9.00e-01	1.24e+00	9.00e-01	1.06e+00	1.05e+00	1.00e+00	2.13e+00	9.00e-01	9.00e-01	9.96e-01	9.00e-01	1.03e+00
	STD	3.01e-02	4.04e-02	1.46e-01	1.82e-11	1.29e-01	8.97e-16	1.66e-02	1.52e-02	3.98e-09	1.30e+00	1.27e-14	8.97e-16	2.58e-02	8.97e-16	1.97e-01
F25	AVG	5.68e+01	1.19e+00	1.57e+02	9.49e-09	1.35e+00	0.00e+00	6.95e+00	1.21e+01	2.86e+01	1.77e+02	1.05e-12	0.00e+00	0.00e+00	0.00e+00	4.55e-15
	STD	1.72e+01	5.91e+00	3.72e+01	1.37e-09	2.75e+00	0.00e+00	2.51e+00	3.45e+00	1.19e+01	1.49e+02	2.79e-12	0.00e+00	0.00e+00	0.00e+00	1.94e-14
F26	AVG	-6.68e+03	-1.26e+04	-9.91e+03	-1.89e+03	-6.26e+03	-1.26e+04	-1.25e+04	-1.16e+04	-1.00e+04	-8.20e+03	-6.00e+03	-1.26e+04	-8.89e+03	-1.26e+04	-1.17e+04
	STD	7.04e+02	7.35e-12	7.15e+02	4.82e+02	7.58e+02	3.09e+01	6.98e+01	1.75e+02	4.78e+02	2.98e+03	4.70e+02	1.21e-02	6.22e+02	8.66e-02	1.16e+03
F27	AVG	4.52e+00	8.19e-02	1.66e+00	2.54e-03	1.69e-01	6.07e-53	3.86e-01	4.94e-01	1.37e+00	9.85e+00	5.88e-09	0.00e+00	8.99e-02	1.08e-143	1.32e-01
	STD	5.84e-01	3.88e-02	4.64e-01	4.16e-03	4.61e-02	3.73e-52	9.26e-02	1.17e-01	4.47e-01	8.34e+00	6.30e-09	0.00e+00	3.03e-02	7.65e-143	8.43e-02
F28	AVG	5.00e-01	2.18e-02	4.98e-01	3.51e-08	1.21e-01	0.00e+00	4.21e-01	3.90e-01	4.98e-01	4.98e-01	6.77e-15	0.00e+00	4.37e-02	0.00e+00	1.09e-01
	STD	6.45e-05	2.21e-02	3.24e-03	1.36e-08	1.19e-01	0.00e+00	6.65e-02	8.64e-02	3.93e-03	8.89e-03	1.54e-14	0.00e+00	3.93e-17	0.00e+00	1.15e-01
F29	AVG	4.10e+01	0.00e+00	2.37e+01	4.39e-02	1.25e-10	1.48e-27	2.77e-01	1.43e-01	3.82e-02	1.07e+02	1.14e-03	0.00e+00	6.04e-46	7.18e-58	3.16e-31
	STD	6.60e+00	0.00e+00	1.17e+01	2.59e-03	5.28e-11	6.33e-27	7.75e-01	5.02e-01	5.21e-02	2.58e+01	1.37e-03	0.00e+00	1.21e-45	3.97e-57	1.17e-30
F30	AVG	4.34e-232	2.71e-201	4.34e-232	-1.00e+00	2.95e-144	-1.00e+00	1.73e-225	2.17e-210	4.34e-232	2.24e-194	-1.00e+00	-1.00e+00	-2.60e-01	-1.00e+00	-8.00e-01
	STD	0.00e+00	0.00e+00	0.00e+00	4.71e-08	2.09e-143	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	4.39e-14	0.00e+00	4.43e-01	0.00e+00	4.04e-01
F31	AVG	2.82e-01	2.98e-02	4.95e-01	8.56e-11	1.70e-02	0.00e+00	2.82e-02	2.95e-02	1.71e-01	4.05e-01	1.67e-14	0.00e+00	0.00e+00	0.00e+00	4.44e-18
	STD	6.97e-02	6.51e-02	7.47e-02	7.37e-12	3.22e-02	0.00e+00	1.16e-02	1.70e-02	4.29e-02	2.52e-01	4.46e-14	0.00e+00	0.00e+00	0.00e+00	3.14e-17

Table 5
Performance of metaheuristic algorithms compared with the CCAA on fixed-dimension problems.

Benchmark		ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F32	AVG	2.57e-12	1.52e-02	6.10e-02	2.03e-01	1.52e-02	7.06e-13	0.00e+00	0.00e+00	0.00e+00	7.12e-02	2.64e-09	4.37e-06	1.22e-14	1.84e-09	1.52e-02
	STD	3.59e-12	1.08e-01	2.09e-01	2.25e-01	1.08e-01	3.51e-12	0.00e+00	0.00e+00	0.00e+00	1.90e-01	1.86e-08	5.41e-06	3.44e-14	2.32e-09	1.08e-01
F33	AVG	3.98e-01	3.98e-01	3.98e-01	4.55e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01
	STD	7.34e-12	3.36e-16	3.36e-16	6.49e-02	3.42e-07	4.22e-08	3.36e-16	3.36e-16	3.36e-16	4.46e-05	7.30e-16	2.77e-05	5.88e-16	1.86e-09	5.38e-07
F34	AVG	8.82e-02	0.00e+00	4.88e-32	2.84e+01	6.73e-01	6.72e-05	6.60e-32	0.00e+00	9.89e-27	1.26e+01	2.34e+00	1.97e-06	3.78e-02	1.02e-03	7.52e-01
	STD	3.79e-01	0.00e+00	1.29e-31	4.54e+00	1.16e+00	1.51e-04	2.53e-31	0.00e+00	6.99e-26	7.46e+01	2.43e+00	4.59e-06	1.35e-02	2.07e-03	6.32e-01
F35	AVG	5.30e-02	0.00e+00	3.00e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	1.50e-01	0.00e+00	1.03e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
F36	AVG	1.04e+00	9.98e-01	1.02e+00	1.15e+01	2.06e+00	1.16e+00	9.98e-01	9.98e-01	9.98e-01	1.35e+00	4.97e+00	1.23e+00	3.35e+00	9.98e-01	1.12e+00
	STD	1.97e-01	1.55e-16	1.41e-01	2.58e+00	2.02e+00	7.29e-01	0.00e+00	0.00e+00	3.17e-17	1.24e+00	2.75e+00	1.65e+00	3.92e+00	1.78e-13	4.76e-01
F37	AVG	3.00e+00	3.00e+00	4.62e+00	1.80e+01	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	4.64e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00
	STD	4.09e-10	7.87e-15	1.15e+01	1.13e+01	5.35e-06	2.39e-08	3.35e-15	4.45e-15	2.69e-15	1.15e+01	1.30e-14	3.22e-06	1.10e-12	1.85e-09	2.16e-06
F38	AVG	-3.86e+00	-3.86e+00	-3.86e+00	-3.54e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.84e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00
	STD	1.33e-09	2.54e-15	3.14e-15	2.37e-01	2.35e-03	7.57e-04	3.13e-15	3.14e-15	3.14e-15	1.09e-01	5.82e-14	3.84e-03	2.42e-15	3.84e-08	2.76e-03
F39	AVG	-3.23e+00	-3.32e+00	-3.28e+00	-2.24e+00	-3.26e+00	-3.18e+00	-3.31e+00	-3.30e+00	-3.27e+00	-3.25e+00	-3.31e+00	-3.10e+00	-3.22e+00	-3.23e+00	-3.26e+00
	STD	4.98e-02	1.88e-09	5.69e-02	4.79e-01	7.60e-02	8.04e-02	3.90e-02	4.61e-02	5.99e-02	5.95e-02	4.17e-02	1.83e-01	5.61e-02	5.13e-02	7.36e-02
F40	AVG	8.83e-10	0.00e+00	1.07e-43	9.16e-01	4.22e-06	3.55e-04	9.48e-55	4.06e-107	3.50e-217	1.96e-03	6.59e-14	2.11e-06	1.03e-17	4.77e-12	9.33e-05
	STD	9.51e-10	0.00e+00	5.93e-43	1.06e+00	3.73e-06	9.34e-04	3.00e-54	1.27e-106	0.00e+00	3.01e-03	6.74e-14	2.55e-06	3.44e-17	7.48e-12	1.05e-04
F41	AVG	-1.03e+00	-1.03e+00	-1.03e+00	-7.49e-01	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00
	STD	1.01e-11	3.28e-14	2.37e-16	2.10e-01	9.47e-09	2.63e-12	2.44e-16	2.31e-16	2.24e-16	3.19e-04	1.10e-15	1.66e-06	5.30e-16	2.70e-11	9.66e-11
F42	AVG	9.13e-04	3.33e-04	4.87e-03	5.37e-03	2.83e-03	3.63e-04	3.26e-04	3.07e-04	1.60e-03	5.64e-04	7.52e-04	3.64e-04	3.07e-04	5.64e-04	4.82e-04
	STD	4.36e-04	1.82e-04	8.32e-03	5.47e-03	6.55e-03	1.78e-04	1.29e-04	2.22e-19	4.80e-03	5.43e-04	2.83e-03	1.33e-04	4.40e-10	2.82e-04	1.40e-04
F43	AVG	-9.64e+00	-1.02e+01	-9.03e+00	-3.21e+00	-9.22e+00	-5.87e+00	-9.85e+00	-9.85e+00	-5.31e+00	-4.96e+00	-5.06e+00	-1.02e+01	-8.42e+00	-1.02e+01	-9.34e+00
	STD	1.54e+00	3.18e-12	2.13e+00	8.30e-01	2.03e+00	1.88e+00	1.21e+00	1.22e+00	3.54e+00	2.98e+00	3.19e-13	5.10e-05	2.44e+00	2.34e-04	1.89e+00
F44	AVG	-9.22e+00	-1.04e+01	-9.59e+00	-3.98e+00	-1.04e+01	-5.82e+00	-1.04e+01	-9.98e+00	-7.22e+00	-6.03e+00	-5.30e+00	-1.04e+01	-9.45e+00	-1.04e+01	-1.04e+01
	STD	2.21e+00	1.39e-12	2.07e+00	8.37e-01	5.90e-04	1.84e+00	7.53e-15	1.46e+00	3.52e+00	3.44e+00	1.05e+00	1.79e-04	2.06e+00	1.07e-04	6.71e-02
F45	AVG	-9.79e+00	-1.05e+01	-1.02e+01	-3.98e+00	-9.89e+00	-6.10e+00	-1.05e+01	-1.05e+01	-7.80e+00	-5.58e+00	-5.13e+00	-1.05e+01	-7.29e+00	-1.05e+01	-8.91e+00
	STD	1.90e+00	6.39e-11	1.30e+00	5.45e-01	2.22e+00	2.09e+00	9.36e-15	8.94e-15	3.55e+00	3.25e+00	4.81e-13	7.80e-05	2.68e+00	1.01e-04	2.96e+00
F46	AVG	-5.00e+01	-5.00e+01	-5.00e+01	-8.12e+00	-5.00e+01	-5.00e+01	-5.00e+01	-5.00e+01	-5.00e+01	-4.65e+01	-5.00e+01	-4.95e+01	-5.00e+01	-5.00e+01	-5.00e+01
	STD	5.93e-02	7.29e-14	5.84e-14	5.83e+00	8.77e-05	8.51e-03	4.30e-14	1.58e-14	3.86e-14	3.22e+00	4.48e-02	2.47e-01	1.48e-13	3.34e-04	3.62e-05
F47	AVG	3.37e-02	4.21e-03	6.25e-03	4.00e+00	2.75e-02	6.34e-02	2.29e-03	2.29e-03	1.01e-02	9.40e+00	1.58e-02	2.76e-02	1.29e-02	1.74e-02	1.38e+01
	STD	5.42e-02	3.18e-03	1.97e-02	2.13e+00	3.97e-02	7.07e-02	1.12e-17	1.34e-17	3.46e-02	2.02e+01	8.66e-03	4.72e-02	5.54e-03	1.01e-02	2.08e+01
F48	AVG	0.00e+00	0.00e+00	0.00e+00	2.97e-06	0.00e+00	0.00e+00	4.70e-120	5.05e-119	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	0.00e+00	0.00e+00	0.00e+00	1.92e-07	0.00e+00	0.00e+00	8.66e-120	1.19e-118	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00

Table 6
Ranking of the compared algorithms on 30-dimensional and fixed-dimensional problems.

Function	ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F1	12	1	11	10	5	3	7	6	9	13	8	1	2	1	4
F2	14	1	13	11	9	3	7	5	12	15	8	4	6	2	10
F3	11	1	12	5	8	2	9	7	10	13	4	1	3	1	6
F4	11	1	8	10	3	4	5	6	7	12	9	1	1	1	2
F5	12	1	10	9	5	3	8	6	11	13	7	1	2	1	4
F6	14	6	13	12	7	4	10	9	11	15	1	2	3	5	8
F7	14	3	13	11	9	1	6	5	12	15	8	4	7	2	10
F8	10	1	12	7	6	3	9	8	11	14	5	1	4	2	13
F9	11	1	10	7	5	3	9	8	14	12	6	1	4	2	13
F10	13	1	12	8	6	5	9	11	10	14	7	1	3	2	4
F11	12	1	11	9	5	3	8	6	10	13	7	1	2	1	4
F12	6	1	5	1	1	1	2	3	4	7	1	1	1	1	1
F13	13	1	12	10	6	4	9	7	11	14	8	1	3	2	5
F14	11	1	10	6	4	2	8	7	9	13	5	1	3	1	12
F15	10	1	12	7	4	1	8	5	9	11	6	1	3	1	2
F16	14	4	12	8	11	3	7	10	9	15	6	2	5	1	13
F17	8	1	7	3	1	1	5	4	6	9	2	1	1	1	1
F18	11	1	10	3	4	1	7	5	9	12	2	1	6	1	8
F19	13	1	12	9	6	4	8	10	11	14	7	1	3	2	5
F20	13	1	14	11	10	3	6	5	12	15	8	2	9	4	7
F21	9	2	13	7	15	4	14	12	11	10	1	3	8	5	6
F22	14	1	11	13	10	3	7	6	12	15	8	2	9	4	5
F23	14	1	10	13	9	3	6	4	11	15	12	2	8	5	7
F24	8	4	11	3	12	1	10	9	6	13	2	1	5	1	7
F25	10	5	11	4	6	1	7	8	9	12	3	1	1	1	2
F26	12	1	9	15	13	4	5	7	8	11	14	2	10	3	6
F27	14	6	13	5	9	3	10	11	12	15	4	1	7	2	8
F28	13	4	11	3	7	1	9	8	12	10	2	1	5	1	6
F29	13	1	12	9	6	5	11	10	8	14	7	1	3	2	4
F30	6	9	6	3	11	1	7	8	6	10	2	1	5	1	4
F31	10	8	12	4	5	1	6	7	9	11	3	1	1	1	2
F32	4	8	11	13	10	3	1	1	1	12	6	7	2	5	9
F33	4	1	1	11	7	6	1	1	1	10	3	9	2	5	8
F34	9	1	2	14	10	6	3	1	4	13	12	5	8	7	11
F35	2	1	3	1	1	1	1	1	1	1	1	1	1	1	1
F36	4	1	3	12	9	6	1	1	1	8	11	7	10	2	5
F37	6	4	12	14	11	8	1	1	2	13	3	10	5	7	9
F38	3	1	1	10	6	5	1	1	1	9	2	8	1	4	7
F39	11	1	5	15	7	13	2	4	6	9	3	14	12	10	8
F40	9	1	5	15	11	13	4	3	2	14	7	10	6	8	12
F41	5	3	1	11	8	4	1	1	1	10	2	9	1	6	7
F42	11	4	14	15	13	5	3	1	12	8	10	6	2	9	7
F43	6	1	9	15	8	11	4	5	12	14	13	2	10	3	7
F44	10	2	8	15	5	13	1	7	11	12	14	4	9	3	6
F45	7	2	5	14	6	11	1	1	9	12	13	4	10	3	8
F46	10	5	4	15	8	11	3	1	2	14	12	13	6	9	7
F47	11	3	4	13	9	12	2	1	5	14	7	10	6	8	15
F48	1	1	1	4	1	1	2	3	1	1	1	1	1	1	1
AVG	9.77	2.33	8.90	9.23	7.25	4.38	5.65	5.35	7.77	11.85	6.10	3.44	4.69	3.15	6.60
RANK	14	1	12	13	10	4	7	6	11	15	8	3	5	2	9

5.3. Analysis of convergence behavior

The convergence of the CCAA is analyzed to show that it performs an efficient optimization action. According to Van Den Bergh and Engelbrecht (2006), a metaheuristic agent must make abrupt changes in the initial stages of the process to carry out an extensive exploration in the search space. These changes should be reduced while the optimization finishes emphasizing the exploitation of target regions. This behavior can guarantee that an SI algorithm will eventually converge to a point in the search space.

In order to observe the characteristics of the CCAA convergence and show that it has an efficient exploration–exploitation behavior, Fig. 6 illustrates the evolution of the CCAA in different unimodal and multimodal test functions in 500 dimensions as well as in functions with fixed dimensions. In this experiment, 50 independent runs were calculated using the parameters described in Table 2, and taking the average values for the dynamics of the 50 calculated best smart-cells. The first column is a sample of the function in 2 dimensions. The second column presents the average movement of the first value in the position of the best smart-cell. The third describes the average Euclidean distance in each step between the previous and the current position of the best smart-cell. The fourth describes the average weighting of the exploration–exploitation process as the optimization progresses as calculated in Fig. 4, and the last shows the average convergence of the best smart-cell fitness value during the optimization process.

It can be seen in Fig. 6 that in all cases, the CCAA tends first to make an extensive exploration in the search space, as evidenced by the abrupt changes of its first position of the smart-cell and the Euclidean distance in the first stages of optimization of the functions. These changes gradually decrease as the algorithm advances in the iterations to favor the exploitation of promising regions in the optimization's last stages. This behavior is due first to the action of the rules with high exploration capacity (R_2, R_4, R_6 and R_9) that relocate smart-cells and improve their positions rapidly, and then, to the rules (R_1, R_3, R_5, R_7, R_8 and R_{10})

Table 7
Results of the Wilcoxon rank-sum test of all the compared algorithms on 30-dimensional and fixed-dimensional problems.

Function	ADIFA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F1	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F2	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F3	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F4	+	+	+	+	+	+	+	+	+	+	≈	≈	≈	+
F5	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F6	+	+	+	+	−	+	+	+	+	−	−	−	−	+
F7	+	+	+	+	−	+	+	+	+	+	+	+	−	+
F8	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F9	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F10	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F11	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F12	+	+	≈	≈	≈	+	+	+	+	≈	≈	≈	≈	≈
F13	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F14	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F15	+	+	+	+	≈	+	+	+	+	+	≈	+	≈	+
F16	+	+	+	+	−	+	+	+	+	+	≈	+	−	+
F17	+	+	+	≈	≈	+	≈	+	+	+	≈	≈	≈	≈
F18	+	+	+	+	≈	+	+	+	+	+	≈	+	≈	+
F19	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F20	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F21	+	+	+	+	≈	+	+	+	+	−	+	+	+	+
F22	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F23	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F24	+	+	−	+	−	+	+	+	+	−	−	+	−	+
F25	+	+	−	+	≈	+	+	+	+	−	≈	≈	≈	≈
F26	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F27	+	+	−	+	−	+	+	+	+	−	−	+	−	≈
F28	+	+	≈	+	−	+	+	+	+	≈	−	≈	−	+
F29	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F30	−	−	−	+	−	−	≈	−	+	−	−	−	−	−
F31	+	+	−	−	−	−	−	+	+	−	−	−	−	−
F32	−	≈	+	+	−	≈	≈	≈	+	−	−	−	−	+
F33	+	≈	+	+	+	≈	≈	≈	+	+	+	+	+	+
F34	+	+	+	+	+	+	≈	+	+	+	+	+	+	+
F35	+	+	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
F36	+	+	+	+	+	≈	≈	≈	+	+	+	+	+	+
F37	+	+	+	+	≈	−	−	−	+	−	+	+	+	+
F38	+	≈	+	+	+	≈	≈	≈	+	+	+	≈	+	+
F39	+	+	+	+	+	+	+	≈	+	+	+	+	+	+
F40	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F41	+	−	+	+	+	−	−	−	+	−	+	≈	+	+
F42	+	+	+	+	+	−	−	+	+	+	+	−	+	+
F43	+	+	+	+	+	+	+	+	+	+	+	≈	+	+
F44	+	+	+	+	+	−	+	≈	+	+	+	+	+	+
F45	+	+	+	+	+	−	−	+	+	+	+	+	+	+
F46	+	−	+	+	+	−	−	−	+	+	+	+	+	+
F47	+	+	+	+	+	−	−	+	+	+	+	+	+	+
F48	≈	≈	+	≈	≈	+	+	≈	≈	≈	≈	≈	≈	≈
DIFF	43	38	35	42	21	25	26	32	46	24	14	28	16	38

focused on the exploitation of promising regions of the search space as the weighting of the exploration–exploitation rules confirms in the fourth column in Fig. 6, verifying the efficient operation of the CCAA for the different test functions. Reviewing the convergence curves, one can see an accelerated decay pattern in all the functions at the beginning of the optimization process, and then focus on the exploitation stage. The following sections verify the performance of the CCAA in engineering problems.

5.4. Engineering design problems

In engineering, there are many problems where mathematical models are applied that are later optimized. To show the utility of the CCAA in these cases, this section tests for 4design problems: a gear train, a pressure vessel, a welded beam, and a cantilever beam.

Given that these problems have different restrictions, the cost function is penalized with an additional extra-large cost (because they are minimization problems) if one of the restrictions is not fulfilled. This objective function increment is simple to implement and has a low computational cost (Mirjalili & Lewis, 2016).

5.4.1. Gear train design (GTD)

The GTD problem was defined in Sandgren (1990) and has no restrictions other than being discrete and that each parameter is within a

Table 8
Performance of metaheuristic algorithms compared with the CCAA on 500-dimensional unimodal problems.

Benchmark		ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F1	AVG	2.68e+04	0.00e+00	6.91e+04	3.45e-08	3.24e-05	2.15e-100	1.17e+03	1.09e+02	2.58e+12	1.25e+21	7.98e-10	0.00e+00	2.01e-85	0.00e+00	2.09e-91
	STD	5.63e+03	0.00e+00	7.94e+04	1.90e-10	2.11e-04	1.52e-99	1.78e+02	3.18e+01	1.32e+13	1.87e+21	7.63e-10	0.00e+00	2.63e-85	0.00e+00	1.95e-91
F2	AVG	1.74e+06	2.46e-01	1.54e+08	1.00e+00	7.64e-01	2.50e-01	1.03e+06	3.32e+04	8.45e+08	7.17e+08	8.05e-01	5.92e-01	6.67e-01	6.59e-01	6.67e-01
	STD	1.39e+05	2.03e-01	2.38e+07	1.37e-06	1.50e-01	2.06e-04	1.12e+05	4.49e+03	2.89e+07	2.80e+08	1.13e-01	1.62e-01	2.92e-16	3.23e-01	4.21e-04
F3	AVG	8.18e+03	0.00e+00	6.49e+04	1.29e-07	6.18e-04	1.09e-105	2.10e+03	3.53e+02	1.13e+05	5.06e+05	7.88e-09	0.00e+00	8.31e-68	2.31e-204	1.44e-89
	STD	1.29e+03	0.00e+00	1.28e+04	2.89e-09	3.50e-04	3.03e-105	4.18e+02	4.80e+01	8.70e+03	2.93e+05	7.57e-09	0.00e+00	6.80e-68	0.00e+00	5.41e-89
F4	AVG	2.09e-07	0.00e+00	2.16e-07	4.40e-13	4.27e-08	2.31e-132	1.56e-18	1.96e-17	2.91e+00	6.08e+01	1.00e-21	0.00e+00	0.00e+00	0.00e+00	2.11e-143
	STD	1.24e-07	0.00e+00	1.92e-07	1.55e-13	1.00e-07	5.67e-132	2.92e-18	4.54e-17	3.58e-01	5.93e+01	1.94e-21	0.00e+00	0.00e+00	0.00e+00	8.86e-143
F5	AVG	1.51e+02	0.00e+00	7.18e+03	6.15e-19	5.13e-13	9.13e-211	6.01e+01	4.51e+01	5.56e+04	8.17e+04	1.50e-18	0.00e+00	4.42e-142	0.00e+00	1.50e-142
	STD	3.81e+01	0.00e+00	2.45e+03	1.67e-20	5.07e-13	0.00e+00	1.33e+01	1.01e+01	6.01e+03	5.25e+04	3.81e-18	0.00e+00	5.24e-142	0.00e+00	2.42e-142
F6	AVG	1.95e+02	1.28e-04	7.95e+03	2.08e-02	1.90e-02	4.90e-05	7.58e+01	4.63e+01	5.75e+04	7.87e+04	4.07e-04	3.25e-05	8.80e-05	7.59e-04	1.57e-03
	STD	4.39e+01	8.60e-05	1.95e+03	1.96e-02	5.22e-03	4.25e-05	1.42e+01	5.54e+00	1.94e+03	5.20e+04	3.32e-04	2.87e-05	1.46e-04	4.77e-04	2.10e-03
F7	AVG	1.80e+07	0.00e+00	9.94e+08	4.99e+02	4.97e+02	3.82e-02	9.64e+06	2.68e+05	6.96e+09	3.31e+09	4.98e+02	3.06e+02	4.95e+02	1.04e+01	4.95e+02
	STD	5.60e+06	0.00e+00	3.07e+08	2.86e-02	1.53e-01	4.46e-02	1.72e+06	3.98e+04	2.85e+08	2.43e+09	1.10e-01	2.42e+02	8.48e-01	1.44e+01	2.60e-01
F8	AVG	5.22e+05	3.14e-09	3.30e+06	3.23e-06	2.18e+05	4.29e-52	2.23e+05	1.54e+05	3.81e+06	2.18e+07	5.61e-08	0.00e+00	7.39e-61	1.31e-30	2.09e+07
	STD	1.40e+05	2.12e-08	3.38e+05	1.94e-06	4.35e+04	2.12e-51	3.66e+04	2.57e+04	4.18e+05	8.42e+06	2.26e-07	0.00e+00	8.12e-62	9.27e-30	4.34e+06
F9	AVG	5.64e+01	8.09e-152	6.76e+01	3.59e-05	6.12e+01	1.25e-51	3.79e+01	2.88e+01	9.93e+01	5.29e+01	1.20e-05	0.00e+00	2.95e-32	2.09e-40	7.71e+01
	STD	6.34e+00	5.72e-151	4.89e+00	2.20e-06	6.78e+00	2.09e-51	2.93e+00	2.86e+00	1.88e-01	2.54e+01	9.96e-06	0.00e+00	5.17e-34	1.03e-39	1.73e+01
F10	AVG	4.71e+290	0.00e+00	9.92e+03	4.27e-03	1.55e+88	8.48e-53	3.11e+192	7.97e+164	6.57e+202	6.71e+292	4.08e-05	2.71e+182	2.48e-57	6.34e-01	1.10e-51
	STD	2.37e+02	0.00e+00	2.07e+03	1.30e-04	1.10e+89	2.49e-52	1.07e+01	1.22e+02	4.03e+01	2.33e+02	4.76e-05	9.13e+01	3.49e-57	9.30e-01	3.25e-51
F11	AVG	6.08e+04	0.00e+00	5.33e+05	5.76e-08	3.22e-05	2.77e-100	3.30e+04	1.26e+04	9.64e+05	8.73e+05	4.85e-10	0.00e+00	3.55e-81	7.60e-100	1.16e-86
	STD	9.70e+03	0.00e+00	3.69e+04	3.77e-09	9.15e-06	1.12e-99	4.67e+03	2.07e+03	1.13e+05	4.80e+05	5.70e-10	0.00e+00	1.37e-80	3.76e-99	6.79e-86
F12	AVG	2.53e+04	0.00e+00	5.07e+05	0.00e+00	2.00e-02	0.00e+00	3.62e+04	1.87e+04	9.15e+05	8.38e+05	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	3.63e+03	0.00e+00	1.04e+05	0.00e+00	1.41e-01	0.00e+00	3.08e+03	3.87e+03	1.25e+05	6.66e+05	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
F13	AVG	1.44e+07	0.00e+00	1.22e+08	1.45e-05	7.87e-03	4.97e-101	6.55e+06	3.29e+06	1.79e+08	2.70e+08	2.64e-07	0.00e+00	1.12e-78	9.14e-16	1.47e-84
	STD	2.35e+06	0.00e+00	2.24e+07	6.89e-07	2.99e-03	1.48e-100	7.77e+05	5.56e+05	2.05e+07	1.24e+08	3.64e-07	0.00e+00	5.09e-78	6.47e-15	9.44e-84
F14	AVG	2.30e+04	3.76e-02	9.85e+15	3.28e-01	3.26e+03	2.85e+03	4.93e+03	2.18e+03	1.27e+04	5.11e+19	6.07e-07	0.00e+00	9.14e-50	2.01e+03	7.83e+03
	STD	1.45e+03	7.99e-02	4.87e+16	4.60e-03	3.71e+02	3.46e+03	7.32e+02	3.77e+02	6.63e+02	6.23e+19	8.68e-07	0.00e+00	5.83e-49	4.24e+03	6.32e+02

Table 9
Performance of metaheuristic algorithms compared with the CCAA on 500-dimensional multimodal problems.

Benchmark		ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F15	AVG	1.32e+01	8.88e-16	2.00e+01	1.62e-05	3.01e-04	8.88e-16	1.27e+01	8.52e+00	2.07e+01	2.01e+01	2.18e-06	8.88e-16	4.09e-15	8.88e-16	5.15e-15
	STD	1.35e+00	0.00e+00	1.65e-04	4.43e-07	4.45e-05	0.00e+00	3.37e-01	4.00e-01	4.15e-01	1.92e+00	2.18e-06	0.00e+00	1.08e-15	0.00e+00	2.69e-15
F16	AVG	3.50e+02	1.34e-11	6.13e+02	7.74e-05	4.12e-02	6.01e-05	1.37e+02	6.58e+01	5.20e+02	9.24e+02	7.10e-07	2.72e-100	4.57e-13	3.25e-03	1.42e-54
	STD	4.27e+01	6.68e-11	7.50e+01	1.61e-06	8.17e-03	3.48e-04	8.42e+00	7.63e+00	5.64e+01	4.99e+02	8.61e-07	1.92e-99	2.70e-12	2.30e-02	4.99e-54
F17	AVG	4.83e+01	0.00e+00	1.01e+02	1.42e-08	3.76e-08	0.00e+00	2.62e+01	2.86e+01	1.77e+02	2.66e+02	5.27e-09	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	3.47e+00	0.00e+00	1.20e+01	1.07e-10	1.26e-08	0.00e+00	1.56e+00	1.46e+00	3.45e+01	7.59e+01	1.05e-08	0.00e+00	0.00e+00	0.00e+00	0.00e+00
F18	AVG	1.54e+01	0.00e+00	1.16e+02	3.70e-10	2.48e-03	0.00e+00	9.13e+00	4.46e+00	2.39e+02	2.09e+02	1.33e-12	0.00e+00	1.97e-04	0.00e+00	0.00e+00
	STD	2.39e+00	0.00e+00	2.57e+01	5.56e-11	1.01e-02	0.00e+00	8.95e-01	6.94e-01	3.10e+01	1.52e+02	1.68e-12	0.00e+00	1.39e-03	0.00e+00	0.00e+00
F19	AVG	3.25e+09	0.00e+00	1.05e+10	3.94e-03	6.96e-02	7.41e-94	2.27e+08	1.46e+08	8.09e+09	5.78e+10	1.51e-06	0.00e+00	6.28e-78	1.44e-04	3.09e-83
	STD	7.53e+08	0.00e+00	2.87e+09	5.96e-04	2.60e-02	5.24e-93	3.69e+07	2.43e+07	1.23e+09	3.58e+10	2.35e-06	0.00e+00	1.08e-77	7.13e-04	1.38e-82
F20	AVG	4.38e+02	1.50e-32	1.50e+03	4.59e+01	3.94e+01	2.06e-04	1.48e+02	2.87e+01	5.70e+03	3.39e+03	3.94e+01	3.91e-06	3.13e+01	4.16e-02	7.91e+00
	STD	8.74e+01	1.38e-47	2.67e+02	5.08e-02	5.90e-01	2.37e-04	1.50e+01	3.25e+00	1.86e+02	2.43e+03	3.87e-01	6.89e-06	1.23e+00	3.04e-02	2.59e+00
F21	AVG	8.72e+01	1.50e-06	1.46e+02	3.23e-02	2.40e+02	4.77e-04	2.30e+02	2.27e+02	2.01e+02	1.59e+02	8.20e-08	3.72e-06	4.45e+01	1.89e-03	2.55e-01
	STD	3.45e+01	3.46e-06	3.77e+00	1.44e-02	7.50e-01	1.45e-03	8.46e-01	2.82e+00	1.04e+01	6.02e+01	1.52e-07	5.63e-06	2.26e+01	2.61e-03	5.97e-01
F22	AVG	1.44e+05	9.42e-34	1.70e+09	1.21e+00	6.58e-01	3.07e-07	7.75e+04	1.43e+01	1.74e+10	7.77e+09	6.67e-01	1.76e-09	2.23e-01	6.81e-04	2.00e-02
	STD	2.35e+05	6.91e-49	8.39e+08	1.31e-03	3.45e-02	4.09e-07	5.11e+04	6.24e+00	7.03e+08	7.71e+09	1.56e-02	2.24e-09	2.57e-02	1.06e-03	8.00e-03
F23	AVG	8.15e+06	1.35e-32	3.98e+09	5.00e+01	4.66e+01	7.86e-05	5.49e+06	9.42e+03	3.19e+10	1.36e+10	4.70e+01	9.89e-01	3.20e+01	2.74e-01	5.91e+00
	STD	5.36e+06	1.11e-47	1.29e+09	7.25e-03	1.10e+00	7.71e-05	2.21e+06	1.14e+04	8.80e+08	1.40e+10	3.83e-01	6.99e+00	8.68e-01	2.86e-01	1.59e+00
F24	AVG	9.53e+00	9.07e-01	7.00e+01	9.00e-01	1.09e+01	9.00e-01	1.41e+02	8.49e+01	4.30e+01	8.31e+01	9.00e-01	9.00e-01	9.52e-01	9.20e-01	9.02e-01
	STD	6.07e+00	2.42e-02	2.13e+01	6.70e-11	2.89e+01	8.97e-16	2.16e+00	4.84e+01	5.89e+00	5.90e+01	9.60e-10	8.97e-16	5.05e-02	4.11e-02	1.53e-02
F25	AVG	3.72e+03	0.00e+00	5.61e+03	2.43e-07	5.05e+01	0.00e+00	3.03e+03	7.44e+02	4.85e+03	6.80e+03	3.57e-08	0.00e+00	0.00e+00	0.00e+00	1.82e-14
	STD	5.32e+02	0.00e+00	6.29e+02	6.24e-09	2.17e+01	0.00e+00	9.19e+02	9.12e+01	3.22e+02	2.88e+03	1.28e-07	0.00e+00	0.00e+00	0.00e+00	1.29e-13
F26	AVG	-3.51e+04	-2.09e+05	-8.72e+04	-7.72e+03	-6.22e+04	-2.09e+05	-5.66e+04	-5.14e+04	-8.26e+04	-5.80e+04	-3.65e+04	-2.09e+05	-1.20e+05	-2.09e+05	-2.06e+05
	STD	3.75e+03	1.18e-10	4.21e+03	2.77e+03	1.09e+04	4.11e+00	1.83e+03	1.52e+03	5.35e+03	5.66e+04	2.65e+03	2.68e-01	1.49e+04	1.97e+01	9.32e+03
F27	AVG	2.62e+01	2.40e-02	8.85e+01	2.45e-04	8.06e-01	6.02e-51	2.90e+01	1.67e+01	1.09e+02	8.90e+01	3.09e-06	0.00e+00	7.99e-02	1.74e-42	1.24e-01
	STD	2.66e+00	4.31e-02	5.41e+00	1.16e-04	8.67e-02	4.22e-50	1.47e+00	1.41e+00	3.85e+00	3.62e+01	2.34e-06	0.00e+00	4.04e-02	1.22e-41	6.56e-02
F28	AVG	5.00e-01	9.61e-03	5.00e-01	4.93e-06	4.97e-01	0.00e+00	5.00e-01	5.00e-01	5.00e-01	5.00e-01	4.58e-10	0.00e+00	3.67e-02	0.00e+00	8.91e-02
	STD	6.60e-08	1.83e-02	4.39e-11	1.71e-06	1.28e-03	0.00e+00	8.24e-09	3.32e-08	1.59e-11	2.77e-03	5.33e-10	0.00e+00	1.62e-02	0.00e+00	1.00e-01
F29	AVG	1.29e+03	0.00e+00	1.83e+03	8.38e-01	8.10e-01	1.80e-26	9.95e+02	7.33e+02	1.82e+03	2.26e+03	4.88e-02	0.00e+00	1.52e-32	8.40e-02	4.46e-31
	STD	5.63e+01	0.00e+00	8.80e+01	1.24e-02	7.45e-02	1.10e-25	3.16e+01	3.22e+01	3.88e+01	2.71e+02	2.04e-02	0.00e+00	3.56e-32	2.49e-01	1.51e-30
F30	AVG	0.00e+00	0.00e+00	0.00e+00	-1.00e+00	0.00e+00	-9.40e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	-1.00e+00	0.00e+00	0.00e+00	-1.00e+00	0.00e+00
	STD	0.00e+00	0.00e+00	0.00e+00	9.88e-08	0.00e+00	2.40e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.21e-09	0.00e+00	0.00e+00	0.00e+00	0.00e+00
F31	AVG	3.83e-01	3.55e-03	7.16e-01	1.14e-10	3.31e-02	0.00e+00	7.72e-01	5.16e-01	6.89e-01	8.95e-01	5.45e-11	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	8.08e-02	2.51e-02	5.07e-02	2.80e-12	8.59e-03	0.00e+00	6.34e-03	1.71e-01	6.70e-02	9.99e-02	9.54e-11	0.00e+00	0.00e+00	0.00e+00	0.00e+00

Table 10
Performance of metaheuristic algorithms compared with the CCAA on fixed-dimensional problems.

Benchmark		ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F32	AVG	4.56e-12	0.00e+00	6.10e-02	1.84e-01	6.99e-08	8.16e-15	0.00e+00	0.00e+00	0.00e+00	8.77e-02	5.48e-11	7.80e-06	9.34e-15	3.38e-10	3.11e-12
	STD	6.65e-12	0.00e+00	2.09e-01	2.28e-01	6.78e-08	3.43e-14	0.00e+00	0.00e+00	0.00e+00	2.41e-01	3.87e-10	1.10e-05	1.86e-14	3.44e-10	5.18e-12
F33	AVG	3.98e-01	3.98e-01	3.98e-01	4.41e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01
	STD	5.26e-12	3.36e-16	3.36e-16	6.05e-02	3.56e-07	1.93e-08	3.36e-16	3.36e-16	3.36e-16	1.04e-04	6.90e-16	2.30e-05	5.08e-16	7.59e-10	1.50e-07
F34	AVG	1.37e-01	0.00e+00	2.13e-32	2.80e+01	7.00e-01	1.48e-04	2.54e-31	0.00e+00	1.38e-31	6.08e+00	1.97e+00	2.07e-06	3.68e-02	2.62e-03	1.05e+00
	STD	5.91e-01	0.00e+00	5.83e-32	5.21e+00	1.25e+00	3.39e-04	1.70e-30	0.00e+00	4.45e-31	2.35e+01	2.45e+00	2.71e-06	1.79e-02	3.75e-03	1.12e+00
F35	AVG	2.15e-01	0.00e+00	6.00e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.26e-02	2.05e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	8.25e-01	0.00e+00	1.39e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.88e-01	9.47e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
F36	AVG	9.98e-01	9.98e-01	1.06e+00	1.19e+01	2.65e+00	1.12e+00	9.98e-01	1.08e+00	9.98e-01	1.04e+00	7.67e+00	9.98e-01	2.25e+00	9.98e-01	1.14e+00
	STD	4.89e-05	1.38e-16	3.11e-01	2.01e+00	2.88e+00	3.26e-01	0.00e+00	3.93e-01	0.00e+00	2.81e-01	3.98e+00	1.05e-10	3.20e+00	2.53e-14	4.02e-01
F37	AVG	3.00e+00	3.00e+00	3.00e+00	1.64e+01	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.04e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00
	STD	3.83e-10	7.87e-15	3.31e-15	1.62e+01	6.48e-06	3.79e-09	3.66e-15	3.45e-15	3.05e-15	1.17e-01	5.42e-14	3.05e-06	7.38e-13	7.44e-11	3.64e-06
F38	AVG	-3.86e+00	-3.86e+00	-3.86e+00	-3.57e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.85e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00
	STD	1.74e-09	9.21e-15	3.14e-15	1.73e-01	2.53e-03	6.03e-04	3.14e-15	3.14e-15	3.14e-15	1.09e-01	4.71e-14	3.47e-03	2.46e-15	1.85e-08	1.22e-03
F39	AVG	-3.23e+00	-3.32e+00	-3.27e+00	-2.38e+00	-3.26e+00	-3.20e+00	-3.29e+00	-3.30e+00	-3.28e+00	-3.23e+00	-3.30e+00	-3.16e+00	-3.30e+00	-3.24e+00	-3.29e+00
	STD	5.27e-02	1.68e-02	5.99e-02	3.52e-01	6.99e-02	7.23e-02	5.27e-02	4.40e-02	5.76e-02	5.11e-02	4.61e-02	9.11e-02	5.15e-02	5.51e-02	5.64e-02
F40	AVG	1.15e-09	0.00e+00	6.96e-41	1.22e+00	4.57e-06	1.53e-04	1.27e-54	1.46e-106	7.67e-214	3.99e-03	6.15e-14	1.53e-06	1.24e-18	1.32e-11	4.56e-05
	STD	1.09e-09	0.00e+00	4.82e-40	1.43e+00	5.08e-06	2.78e-04	4.65e-54	6.54e-106	0.00e+00	7.11e-03	6.50e-14	1.71e-06	4.62e-18	1.65e-11	7.87e-05
F41	AVG	-1.03e+00	-1.03e+00	-1.03e+00	-8.19e-01	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00
	STD	7.88e-12	1.69e-13	2.24e-16	1.70e-01	8.42e-09	4.22e-12	2.50e-16	2.24e-16	2.24e-16	2.24e-04	1.17e-15	1.06e-06	5.65e-16	1.91e-11	1.62e-11
F42	AVG	7.59e-04	3.54e-04	5.67e-03	4.99e-03	4.72e-03	3.89e-04	3.07e-04	3.07e-04	2.01e-03	1.34e-03	4.03e-03	6.56e-04	3.07e-04	6.16e-04	6.93e-04
	STD	4.41e-04	2.21e-04	8.80e-03	4.24e-03	8.39e-03	2.60e-04	2.48e-19	2.15e-19	5.48e-03	3.97e-03	7.75e-03	4.43e-04	7.00e-10	2.74e-04	3.39e-04
F43	AVG	-9.32e+00	-1.02e+01	-9.39e+00	-3.21e+00	-9.24e+00	-5.56e+00	-9.95e+00	-9.85e+00	-5.86e+00	-6.09e+00	-5.06e+00	-1.02e+01	-5.97e+00	-1.02e+01	-9.65e+00
	STD	1.88e+00	3.65e-13	1.94e+00	7.04e-01	1.96e+00	1.53e+00	1.00e+00	1.22e+00	3.49e+00	3.35e+00	2.72e-13	1.77e-04	1.98e+00	1.30e-04	1.76e+00
F44	AVG	-8.66e+00	-1.04e+01	-9.72e+00	-3.98e+00	-1.03e+01	-5.72e+00	-1.04e+01	-1.02e+01	-7.23e+00	-5.31e+00	-5.09e+00	-1.04e+01	-9.23e+00	-1.04e+01	-9.15e+00
	STD	2.49e+00	6.50e-13	1.90e+00	5.84e-01	7.52e-01	1.73e+00	7.57e-15	1.05e+00	3.63e+00	3.29e+00	3.13e-13	1.30e-04	2.22e+00	9.83e-05	2.57e+00
F45	AVG	-9.43e+00	-1.05e+01	-1.01e+01	-3.98e+00	-1.04e+01	-5.87e+00	-1.05e+01	-1.05e+01	-7.42e+00	-5.79e+00	-5.24e+00	-1.05e+01	-9.24e+00	-1.05e+01	-9.28e+00
	STD	2.24e+00	2.40e-13	1.48e+00	6.30e-01	1.15e+00	2.08e+00	9.13e-15	8.95e-15	3.72e+00	3.40e+00	7.65e-01	3.48e-05	2.33e+00	1.00e-04	2.43e+00
F46	AVG	-5.00e+01	-5.00e+01	-5.00e+01	-7.75e+00	-5.00e+01	-5.00e+01	-5.00e+01	-5.00e+01	-5.00e+01	-4.66e+01	-5.00e+01	-4.96e+01	-5.00e+01	-5.00e+01	-5.00e+01
	STD	2.63e-02	5.54e-14	5.72e-14	6.27e+00	1.03e-04	9.33e-03	3.99e-14	1.73e-14	7.03e-14	2.21e+00	2.36e-02	2.26e-01	2.37e-13	1.20e-04	5.11e-05
F47	AVG	4.24e-02	4.85e-03	3.49e-02	3.73e+00	1.29e-01	6.18e-02	2.29e-03	2.29e-03	3.73e-03	5.67e+00	1.52e-02	1.66e-02	1.28e-02	1.65e-02	1.05e+01
	STD	9.52e-02	3.07e-03	9.89e-02	1.86e+00	4.99e-01	4.27e-02	1.28e-17	1.23e-17	6.71e-03	1.30e+01	5.76e-03	8.51e-03	5.56e-03	2.03e-02	1.62e+01
F48	AVG	0.00e+00	0.00e+00	0.00e+00	2.98e-06	0.00e+00	0.00e+00	1.69e-119	5.75e-119	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	STD	0.00e+00	0.00e+00	0.00e+00	1.91e-07	0.00e+00	0.00e+00	6.14e-119	1.08e-118	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00

Table 11
Ranking of the compared algorithms on 500-dimensional and fixed-dimensional problems.

Function	ADIFA	CCAA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F1	10	1	11	6	7	2	9	8	12	13	5	1	4	1	3
F2	12	1	13	9	7	2	11	10	15	14	8	3	5	4	6
F3	11	1	12	7	8	3	10	9	13	14	6	1	5	2	4
F4	9	1	10	7	8	3	5	6	11	12	4	1	1	1	2
F5	10	1	11	5	7	2	9	8	12	13	6	1	4	1	3
F6	12	4	13	9	8	2	11	10	14	15	5	1	3	6	7
F7	12	1	13	9	7	2	11	10	15	14	8	4	5	3	6
F8	11	5	12	7	9	3	10	8	13	15	6	1	2	4	14
F9	11	2	13	7	12	3	9	8	15	10	6	1	5	4	14
F10	11	1	8	6	9	3	11	10	11	11	5	11	2	7	4
F11	11	1	12	7	8	2	10	9	14	13	6	1	5	3	4
F12	4	1	6	1	2	1	5	3	8	7	1	1	1	1	1
F13	11	1	12	7	8	2	10	9	13	14	6	1	4	5	3
F14	13	4	14	5	9	8	10	7	12	15	3	1	2	6	11
F15	9	1	10	5	6	1	8	7	12	11	4	1	2	1	3
F16	12	4	14	7	9	6	11	10	13	15	5	1	3	8	2
F17	7	1	8	3	4	1	5	6	9	10	2	1	1	1	1
F18	8	1	9	3	5	1	7	6	11	10	2	1	4	1	1
F19	11	1	13	7	8	2	10	9	12	14	5	1	4	6	3
F20	12	1	13	10	8	3	11	6	15	14	9	2	7	4	5
F21	9	2	10	6	15	4	14	13	12	11	1	3	8	5	7
F22	12	1	13	9	7	3	11	10	15	14	8	2	6	4	5
F23	12	1	13	9	7	2	11	10	15	14	8	4	6	3	5
F24	8	5	11	3	9	1	14	13	10	12	2	1	7	6	4
F25	8	1	10	4	5	1	7	6	9	11	3	1	1	1	2
F26	14	1	7	15	9	3	11	12	8	10	13	2	6	4	5
F27	11	6	13	5	9	2	12	10	15	14	4	1	7	3	8
F28	9	4	12	3	7	1	11	10	13	8	2	1	5	1	6
F29	11	1	13	8	7	4	10	9	12	14	5	1	2	6	3
F30	5	5	5	3	5	4	5	5	5	5	2	5	5	1	5
F31	6	4	9	3	5	1	10	7	8	11	2	1	1	1	1
F32	5	1	10	12	8	2	1	1	1	11	6	9	3	7	4
F33	4	1	1	11	8	6	1	1	1	10	3	9	2	5	7
F34	9	1	2	14	10	6	4	1	3	13	12	5	8	7	11
F35	3	1	5	1	1	1	1	1	2	4	1	1	1	1	1
F36	4	1	6	13	11	8	1	7	1	5	12	3	10	2	9
F37	8	4	3	14	12	9	1	2	3	13	5	11	6	7	10
F38	4	2	1	11	8	6	1	1	1	10	3	9	1	5	7
F39	11	1	8	15	9	13	6	2	7	12	3	14	4	10	5
F40	9	1	5	15	11	13	4	3	2	14	7	10	6	8	12
F41	5	3	1	11	8	4	1	1	1	10	2	9	1	6	7
F42	9	4	15	14	13	5	2	1	11	10	12	7	3	6	8
F43	8	1	7	15	9	13	4	5	12	10	14	2	11	3	6
F44	10	2	7	15	5	12	1	6	11	13	14	3	8	4	9
F45	7	2	6	14	5	11	1	1	10	12	13	3	9	4	8
F46	10	5	3	15	8	11	2	1	4	14	12	13	6	9	7
F47	10	4	9	13	12	11	2	1	3	14	6	8	5	7	15
F48	1	1	1	4	1	1	2	3	1	1	1	1	1	1	1
AVG	8.94	2.08	9.02	8.38	7.77	4.38	6.96	6.29	9.29	11.54	5.79	3.65	4.33	4.08	5.73
RANK	12	1	13	11	10	5	9	8	14	15	7	2	4	3	6

possible range of values. In this problem, we have 4decision parameters, $\boldsymbol{y} = [y(1), y(2), y(3), y(4)]$, which represent the number of teeth that a gear can have (Fig. 7),Fig. 8,Fig. 9.

The goal of this problem is to determine the minimum cost of gear ratio that can be set as follows:

$$min\, f(\boldsymbol{y}) = \left(\frac{1}{6.931} - \frac{y(2)y(3)}{y(1)y(4)}\right)^2 \tag{1}$$

where $12 \leqslant y(i) \leqslant 60$ for $i = 1, \ldots, 4$. and each $y(i)$ must be integer.

For this case, the result obtained by the CCAA is compared with the results published in Gupta et al. (2020), which takes 17 different metaheuristic algorithms. Fifty independent runs of the CCAA were made and the best result was taken. In each run, $n_S = 5$and $n_{ne} = 4$ were used, in addition to 200evaluations of f to make the setting of the experiment similar to that of the one reported in Gupta et al. (2020). The reference values and the best design obtained by the CCAA are shown in Table 13. It can be seen that the CCAA obtained a design with a similar cost to the MSCA and the FA and a better result compared to the other algorithms. These outcomes show that the CCAA is competitive with the best algorithms for this discrete design problem.

5.4.2. *Pressure vessel design (PVD)*

For this problem, a mathematical model calculates the total cost of a cylindrical pressure vessel, showing the cost relationships concerning the material, the structure, and the weld. The design variables $\boldsymbol{y}$ to

Table 12
Results of the Wilcoxon rank-sum test and ranking of all the compared algorithms on 500-dimensional and fixed-dimensional problems.

Function	ADIFA	CPSO	EHO	GWO	HHO	LSHADE	LSPACMA	MA	MBO	MSA	MSCA	RDWOA	SMA	WOA
F1	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F2	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F3	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F4	+	+	+	+	+	+	+	+	+	+	≈	≈	≈	+
F5	+	+	+	+	+	+	+	+	+	+	≈	+	≈	+
F6	+	+	+	+	−	+	+	+	+	+	−	−	+	+
F7	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F8	+	+	+	+	−	+	+	+	+	+	−	−	−	+
F9	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F10	≈	+	+	+	+	≈	+	≈	+	+	≈	+	+	+
F11	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F12	+	+	≈	≈	≈	+	+	+	+	≈	≈	≈	≈	≈
F13	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F14	+	+	+	+	+	+	+	+	+	−	−	−	+	+
F15	+	+	+	+	≈	+	+	+	+	+	≈	+	≈	+
F16	+	+	+	+	+	+	+	+	+	+	≈	−	+	−
F17	+	+	+	+	≈	+	+	+	+	+	≈	≈	≈	≈
F18	+	+	+	+	≈	+	+	+	+	+	≈	≈	≈	≈
F19	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F20	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F21	+	+	+	+	≈	+	+	+	+	≈	+	+	+	+
F22	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F23	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F24	+	+	−	+	−	+	+	+	+	−	−	+	≈	≈
F25	+	+	+	+	≈	+	+	+	+	+	≈	≈	≈	≈
F26	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F27	+	+	−	+	−	+	+	+	+	−	−	+	−	+
F28	+	+	−	+	−	+	+	+	+	−	−	+	−	+
F29	+	+	+	+	+	+	+	+	+	+	≈	+	+	+
F30	≈	≈	−	≈	−	≈	≈	≈	≈	−	≈	≈	−	≈
F31	+	+	−	+	≈	+	+	+	+	−	≈	≈	≈	≈
F32	+	+	+	+	+	≈	≈	≈	+	+	+	+	+	+
F33	+	≈	+	+	+	≈	≈	≈	+	+	+	≈	+	+
F34	+	+	+	+	+	≈	≈	+	+	+	+	+	+	+
F35	+	+	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
F36	+	+	+	+	+	≈	+	≈	+	+	+	+	+	+
F37	+	−	+	+	≈	−	−	−	+	≈	+	≈	+	+
F38	+	−	+	+	+	−	−	−	+	+	+	−	+	+
F39	+	≈	+	+	+	+	+	+	+	+	+	+	+	+
F40	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F41	+	−	+	+	+	−	−	−	+	−	+	≈	+	+
F42	+	≈	+	+	+	−	−	+	+	+	+	−	+	+
F43	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F44	+	+	+	+	+	−	+	≈	+	+	+	+	+	+
F45	+	+	+	+	+	−	−	≈	+	+	+	≈	+	+
F46	+	−	+	+	+	−	−	−	+	+	+	+	+	+
F47	+	+	+	+	+	−	−	−	+	+	+	+	+	+
F48	≈	≈	+	≈	≈	+	+	≈	≈	≈	≈	≈	≈	≈
DIFF	45	35	36	44	26	25	29	29	45	29	16	23	28	37

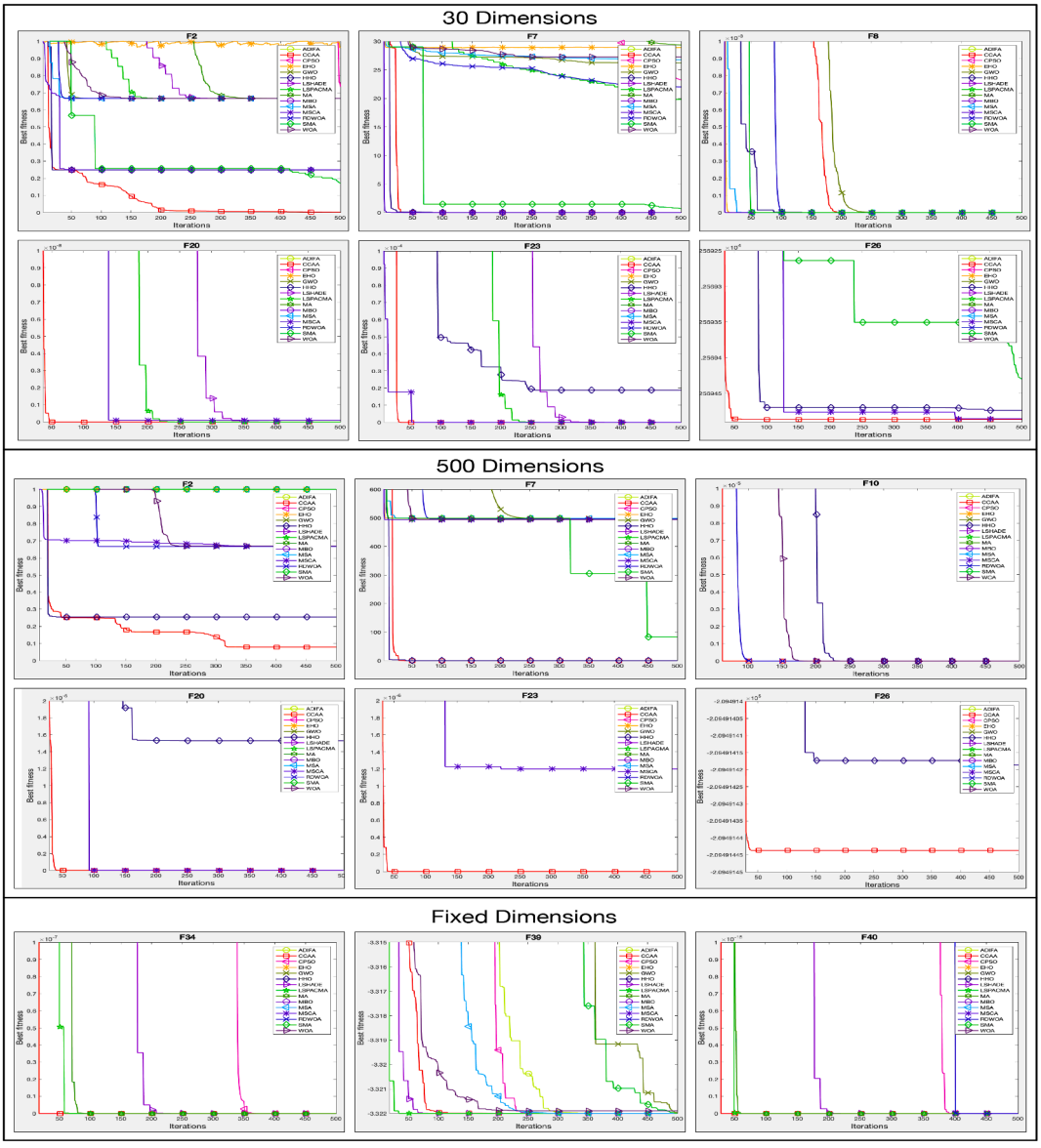

Fig. 5. Convergence curves of the different algorithms for test functions in 30 dimensions.

specify are the thickness of the hull $y(1)$and the head $y(2)$, the radius of the vessel $y(3)$, and the length of the vessel without counting the head $y(4)$, to minimize the total cost of the design (Gandomi et al., 2013). The cost function and its restrictions are shown in Eq. 2.

$$\begin{aligned} &minf(\mathbf{y}) = 0.6224y(1)y(3)y(4) + 1.7781y(1)^2y(3) + 3.1661y(1)^2y(4) \\ &\qquad + 19.84y(1)^2y(3) \\ &\text{Subject to:} \\ &-y(1) + 0.0193y(3) \leqslant 0 \\ &-y(3) + 0.00954y(3) \leqslant 0 \\ &-\pi y(3)^2y(4) - \frac{4}{3}\pi y(3)^3 + 1296000 \leqslant 0 \\ &y(4) - 240 \leqslant 0 \end{aligned} \tag{2}$$

where $0 \leqslant y(1), y(2) \leqslant 99$ and $10 \leqslant y(3), y(4) \leqslant 200$. According to Gandomi et al. (2013), the thicknesses $y(1)$ and $y(2)$ must be multiples of 0.0625 inches because the steel gauges are commercial ones. For this model, the results obtained by the CCAA is compared with the results published in Chen et al. (2020), which considers 7different metaheuristic algorithms. Notice that the best result reported in Chen et al. (2020) does not hold the steel gauge restriction for $y(1)$ and $y(2)$. Therefore, for this case, two different experiments were conducted using the CCAA: one with no gauge restrictions over $y(1)$ and $y(2)$ and the second taking into account that $y(1)$ and $y(2)$ are multiples of 0.0625. Again, 50independent runs of the CCAA were made for each experiment and the best result was taken in every case. In each run, $n_S = 6$and $n_{ne} = 10$ were used, in addition to 15000evaluations of $f$to make an experiment similar to the one reported in Gandomi et al. (2013) and Chen et al., 2020. The reference values and

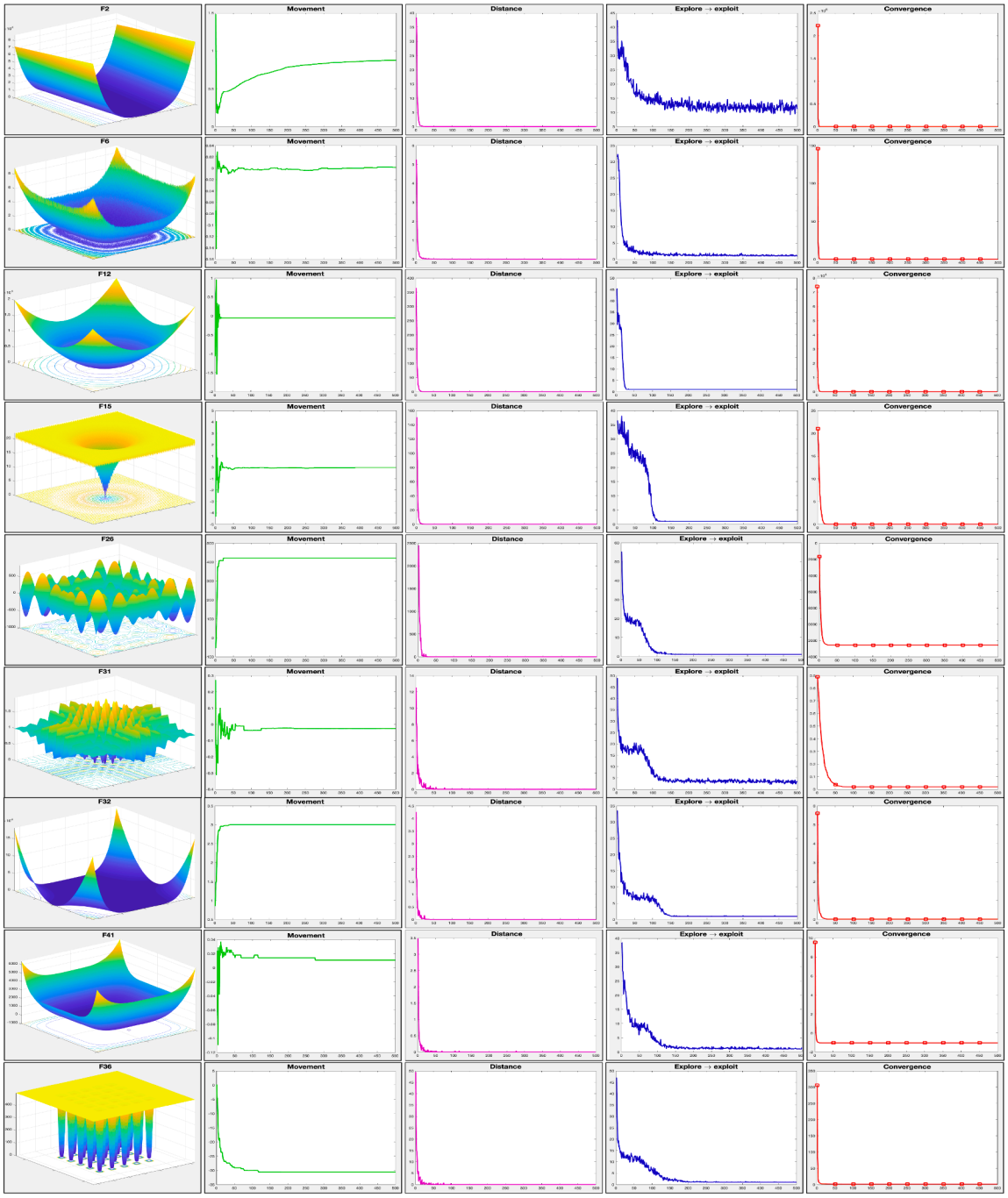

Fig. 6. Average of movement, Euclidean distance, weighting of the exploration–exploitation rules and convergence of the best smart-cell in the CCAA.

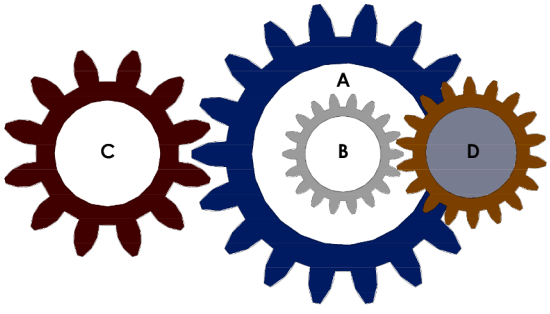

Fig. 7. Gear train.

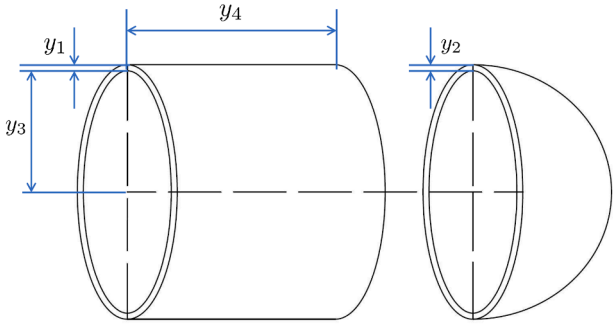

Fig. 8. Pressure vessel.

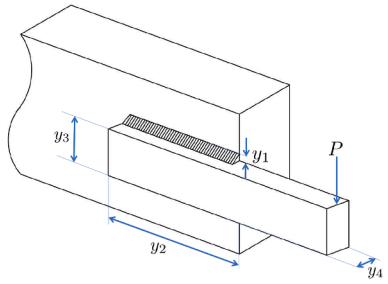

Fig. 9. Welded beam.

Table 13
Comparison of metaheuristics for the GTD problem.

Algorithm	$y(1)$	$y(2)$	$y(3)$	$y(4)$	f_{min}
CCAA	49	19	16	43	2.7009$e-12$
MSCA	49	16	19	43	2.7009$e-12$
FA	43	19	16	49	2.7009$e-12$
m-SCA	34	20	13	53	2.3078$e-11$
TLBO	53	30	13	51	2.3078$e-11$
SCA-GWO	51	26	15	53	2.3078$e-11$
CMA-ES	53	13	20	34	2.3078$e-11$
wPSO	57	12	37	54	8.8876$e-10$
ISCA	46	26	12	47	9.9216$e-10$
PSO	47	13	12	23	9.9216$e-10$
GWO	23	13	12	47	9.9216$e-10$
SCA-PSO	46	24	13	47	9.9216$e-10$
SCA	60	17	28	55	1.3616$e-09$
SinDE	30	17	14	55	1.3616$e-09$
modSCA	60	18	25	52	2.3576$e-09$
SSA	57	27	17	60	3.6358$e-09$
OBSCA	43	12	31	60	8.7008$e-09$

the best design obtained by the CCAA are shown in Table 14. It can be seen that the CCAA obtained the best design with a cost of 5885.5328 in the case of no steel-gauge restriction. The CCAA also calculated the best design with a cost of 6059.7144, applying the steel-gauge restriction. Therefore, the CCAA can be very useful for the PVD problem.

5.4.3. Welded beam design (WBD)

The objective of the WBD problem is to obtain the minimum manufacturing cost (Coello, 2000) subject to the constants of shear stress τ, bending stress σ, buckling load γ, and deflection δ on the beam. This model has 4 design variables $\boldsymbol{y} = [y(1), y(2), y(3), y(4)]$ and 7 restrictions. The variables are the thickness of welds $y(1)$, the length of clamped bar $y(2)$, the height of the beam $y(3)$, and the thickness of the beam $y(4)$. The optimization model is presented in Eq. 3.

$$min\, f(\boldsymbol{y}) = 1.10471 y(1)^2 + 0.04811 y(3) y(4)(14.0 + y(2))$$

Subjectto :

$$g_1(\boldsymbol{y}) = \tau(\boldsymbol{y}) - \tau_{max} \leqslant 0 \qquad g_2(\boldsymbol{y}) = \sigma(\boldsymbol{y}) - \sigma_{max} \leqslant 0$$
$$g_3(\boldsymbol{y}) = \delta(\boldsymbol{y}) - \delta_{max} \leqslant 0 \qquad g_4(\boldsymbol{y}) = y(1) - y(4) \leqslant 0$$
$$g_5(\boldsymbol{y}) = P - \gamma(\boldsymbol{y}) \leqslant 0 \qquad g_6(\boldsymbol{y}) = 0.125 - y(1) \leqslant 0$$
$$g_7(\boldsymbol{y}) = 0.10471 y(1)^2 + 0.04811 y(3) y(4)(14.0 + y(2)) - 5.0 \leqslant 0$$

where :

$$\mathrm{P} = 6000\ \mathrm{lb},\ \mathrm{L} = 14\ \mathrm{in},\ \sigma_{max} = 30000\ \mathrm{psi},$$
$$\delta_{max} = 0.25\ \mathrm{in},\ \mathrm{E} = 30e6\ \mathrm{psi},\ G = 12e6\ \mathrm{psi}$$

$$M = \mathrm{P}\left(\mathrm{L} + \frac{y(2)}{2}\right),\ R = \sqrt{\frac{y(2)^2}{4} + \left(\frac{y(1) + y(3)}{2}\right)^2} \tag{3}$$

$$J = 2\left(\sqrt{2}\, y(1) y(2)\left(\frac{y(2)^2}{12} + \left(\frac{y(1) + y(3)}{2}\right)^2\right)\right)$$

$$\tau' = \frac{\mathrm{P}}{\sqrt{2} y(1) y(2)},\ \tau'\prime = \frac{MR}{J}$$

$$\tau(\boldsymbol{y}) = \sqrt{(\tau')^2 + 2\tau'\tau'\prime \frac{y(2)}{2R} + (\tau'\prime)^2},\ \sigma(\boldsymbol{y}) = \frac{6\mathrm{PL}}{y(3)^2 y(4)}$$

$$\delta(\boldsymbol{y}) = \frac{4\mathrm{PL}^3}{\mathrm{E} y(3)^3 y(4)},\ \gamma(\boldsymbol{y}) = \frac{4.013\mathrm{E}\sqrt{\frac{y(3)^2 y(4)^6}{36}}}{\mathrm{L}^2}\left(1 - \frac{y(3)}{2\mathrm{L}}\sqrt{\frac{\mathrm{E}}{4\mathrm{G}}}\right)$$

with $0.1 \leqslant y(1) \leqslant 2$, $0.1 \leqslant y(2) \leqslant 10$, $0.1 \leqslant y(3) \leqslant 10$y $0.1 \leqslant y(4) \leqslant 2$. For this model, the result obtained by the CCAA is compared with the results

Table 15
Comparison of metaheuristics for the WBD problem.

Algorithm	$y(1)$	$y(2)$	$y(3)$	$y(4)$	F_{min}
RDWOA	0.205618	3.252958	9.04447	0.20569	1.6961
CCAA	0.20573	3.4705	9.03662	0.20573	1.7248
IHS	0.20573	3.47049	9.03662	0.20573	1.7248
RO	0.203687	3.528467	9.004233	0.207241	1.735344
GA-Coello	0.2088	3.4205	8.9975	0.2100	1.7483
HS	0.2442	6.2231	8.2915	0.2433	2.3807
David	0.2434	6.2552	8.2915	0.2444	2.3841
Simple	0.2792	5.6256	7.7512	0.2796	2.5307
Random	0.4575	4.7313	5.0853	0.6600	4.1185

Table 14
Comparison of metaheuristics for the PVD problem.

Algorithm	$y(1)$	$y(2)$	$y(3)$	$y(4)$	f_{min}
CCAA - no gauge restriction	0.7781858	0.3846655	40.31985	199.9995	5885.5328
RDWOA	0.793769	0.39236	41.127973	189.045124	5912.53868
CCAA – gauge restriction	0.8125	0.4375	42.09845	176.6366	6059.7144
ES	0.8125	0.4375	42.098087	176.640518	6059.7456
PSO	0.8125	0.4375	42.091266	176.7465	6061.0777
GA	0.9375	0.5	48.329	112.679	6410.3811
IHS	1.125	0.625	58.29015	43.69268	7197.73
Lagrangian multiplier	1.125	0.625	58.291	43.69	7198.0428
Branch-and-bound	1.125	0.625	47.7	117.71	8129.1036

published in Chen et al. (2020) and Coello, 2000, which take into consideration 8 different metaheuristic algorithms. Again, 50 independent runs of the CCAA were made and the best result was taken. In each run, $n_S = 5$ and $n_{ne} = 4$ were used, in addition to 2000evaluations of f in order to make the experiment comparable to the references consulted. The best design obtained by the CCAA are shown in Table 15. It can be seen that the CCAA calculated the second-best design at the cost of 1.7248. Therefore, the CCAA is competitive in the optimization of the WBD problem.

5.4.4. Cantilever beam design (CBD)

The cantilever beam design problem (CBD) objective is to obtain the minimum weight of a rigid structural element supported only on one side by another vertical element (Mirjalili, 2015). The model consists of five hollow elements that have a square section. There is a vertical load in the last part of the structure, and there is a vertical displacement restriction. The system to be designed can be seen in Fig. 10.

The problem is formulated in Eq. 4.

$$\begin{aligned} &min\, f(\mathbf{y}) = 0.6224(y(1)+y(2)+y(3)+y(4)+y(5)) \\ &\text{Subject to:} \\ &g(\mathbf{y}) = \frac{61}{y(1)^3}+\frac{37}{y(2)^3}+\frac{19}{y(3)^3}+\frac{7}{y(4)^3}+\frac{1}{y(5)^3} \leqslant 1 \end{aligned} \tag{4}$$

where $0.01 \leqslant y(i) \leqslant 100$ for $i = 1,\ldots,5$. For this model, the result obtained by the CCAA is compared with the results published in Wu et al. (2020) and Mirjalili (2015) taking 7 different metaheuristic algorithms. Again, 50 independent runs of the CCAA were made and the best result was taken. In each run, $n_S = 5$ and $n_{ne} = 4$ were used, with a maximum of 12000evaluations of f to have an experiment similar to the one reported in Mirjalili (2015).

The reference values and the best design achieved by the CCAA are shown in Table 16. It can be seen that the CCAA had the second-best design with a cost of 1.33996. Therefore, the CCAA is also well-suited to optimize this engineering problem.

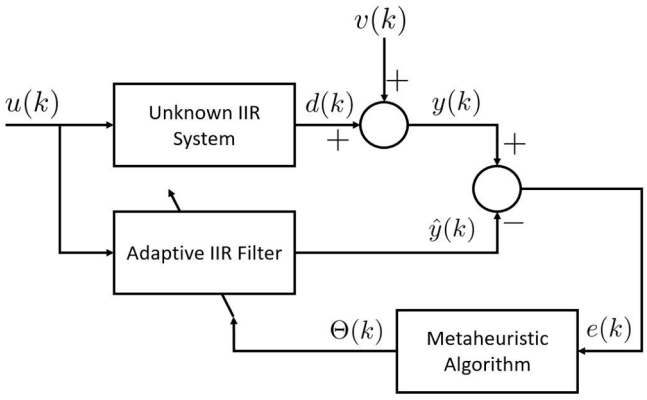

Fig. 11. Parameter identification scheme in adaptive IIR filters.

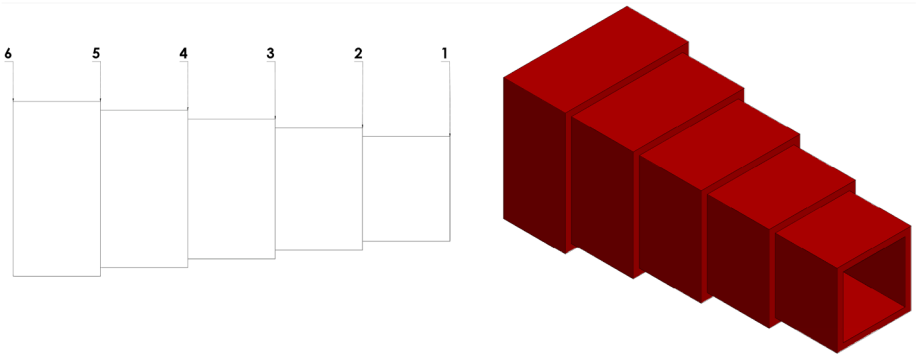

Fig. 10. Cantilever beam.

Table 16
Comparison of metaheuristics for the CBD problem.

Algorithm	$y(1)$	$y(2)$	$y(3)$	$y(4)$	$y(5)$	F_{min}
ALO	6.01812	5.31142	4.48836	3.49751	2.158329	1.33995
CCAA	6.01698	5.30917	4.49428	3.50147	2.15266	1.33996
SOS	6.01878	5.30344	4.49587	3.49896	2.15564	1.33996
CS	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999
MMA	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
GCA_I	6.0100	5.30400	4.4900	3.4980	2.1500	1.3400
GCA_II	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400

Table 17
Ten IIR system identification problems.

Problems	The transfer function of IIR plant $H_p(z)$	The transfer function of the adaptive IIR filter model $H_a(z)$
Example I	$\frac{0.05-0.4z^{-1}}{1-1.1314z^{-1}+0.25z^{-2}}$	$\frac{b_0-b_1z^{-1}}{1-a_1z^{-1}-a_2z^{-2}}$
Example II	$\frac{-0.2-0.4z^{-1}+0.5z^{-2}}{1-0.6z^{-1}+0.25z^{-2}-0.2z^{-3}}$	$\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1-a_1z^{-1}-a_2z^{-2}-a_3z^{-3}}$
Example III	$\frac{1-0.9z^{-1}+0.81z^{-2}-0.729z^{-3}}{1+0.04z^{-1}+0.2775z^{-2}-0.2101z^{-3}+0.14z^{-4}}$	$\frac{b_0+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}}{1-a_1z^{-1}-a_2z^{-2}-a_3z^{-3}-a_4z^{-4}}$
Example IV	$\frac{0.1084+0.5419z^{-1}+1.0837z^{-2}+1.0837z^{-3}+0.5419z^{-4}+0.1084z^{-5}}{1+0.9853z^{-1}+0.9738z^{-2}-0.3854z^{-3}+0.1112z^{-4}+0.0113z^{-5}}$	$\frac{b_0+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}+b_4z^{-4}+b_5z^{-5}}{1-a_1z^{-1}-a_2z^{-2}-a_3z^{-3}-a_4z^{-4}-a_5z^{-5}}$
Example V	$\frac{1-0.4z^{-2}-0.65z^{-4}+0.26z^{-6}}{1-0.77z^{-2}-0.8498z^{-4}+0.6486z^{-6}}$	$\frac{b_0+b_2z^{-2}+b_4z^{-4}+b_6z^{-6}}{1-a_2z^{-2}-a_4z^{-4}-a_6z^{-6}}$
Example VI	$\frac{1}{1-1.4z^{-1}+0.49z^{-2}}$	$\frac{b_0}{1-a_1z^{-1}-a_2z^{-2}}$
Example VII	$\frac{1}{1-1.2z^{-1}+0.6z^{-2}}$	$\frac{b_0}{1-a_1z^{-1}-a_2z^{-2}}$
Example VIII	$\frac{1.25z^{-1}-0.25z^{-2}}{1-0.3z^{-1}+0.4z^{-2}}$	$\frac{b_1z^{-1}+b_2z^{-2}}{1-a_1z^{-1}-a_2z^{-2}}$
Example IX	$\frac{1}{(1-0.5z^{-1})^3}$	$\frac{b_0}{1-a_1z^{-1}-a_2z^{-2}-a_3z^{-3}}$
Example X	$\frac{-0.3+0.4z^{-1}-0.5z^{-2}}{1-1.2z^{-1}+0.5z^{-2}-0.1z^{-3}}$	$\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1-a_1z^{-1}-a_2z^{-2}-a_3z^{-3}}$

6. Infinite-impulse response filter design optimization

Infinite-impulse response (IIR) filters are one of the main types of filters applied in digital signal processing to separate overlapping signals and restore signals that have been distorted (Harris, 2004). This section reports the design of adaptive IIR filters using the CCAA algorithm and comparing the results with other specialized algorithms recently published.

The design of IIR filters has become a recurring research topic since they are more efficient than other types of filters. The IIR filter design problem has been addressed using metaheuristic algorithms, determining the parameters to obtain an optimal model of the unknown plant, minimizing the error's cost function. The IIR filter transfer function is defined by:

Table 18
Experimental results (MSE) of algorithms for Test Examples I,II, III, IV, V.

Problems	Measure	Mean square error (MSE)					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
Example I	Best	2.305e-22	0	1.249e-23	1.237e-07	1.914e-05	0
	Worst	1.006e-20	0	2.504e-01	1.192e-02	1.124e-01	0
	Average	3.384e-21	0	9.334e-03	2.389e-03	3.184e-03	0
	Median	2.524e-21	0	2.734e-18	3.807e-06	2.867e-04	0
	SD	2.348e-21	0	4.632e-02	4.215e-03	1.592e-02	0
Example II	Best	1.118e-21	0	1.914e-32	5.788e-08	2.865e-04	0
	Worst	5.902e-03	1.222e-32	6.121e-03	4.858e-03	3.943e-02	5.589e-03
	Average	1.453e-03	6.079e-34	1.424e-03	1.210e-03	6.084e-03	1.123e-04
	Median	4.134e-21	0	4.572e-04	2.451e-07	4.250e-03	0
	SD	2.092e-03	2.221e-33	1.865e-03	1.643e-03	7.301e-03	7.904e-04
Example III	Best	9.300e-21	0	8.728e-12	1.242e-05	1.648e-03	1.528e-14
	Worst	2.594e-02	1.082e-31	2.794e+00	1.026e-01	3.384e-01	4.980e-02
	Average	3.288e-03	8.660e-33	1.270e-01	1.173e-02	5.761e-02	3.321e-03
	Median	1.566e-13	0	1.206e-03	1.357e-03	2.416e-02	5.887e-05
	SD	7.083e-03	1.852e-32	4.351e-01	2.357e-02	6.200e-02	9.963e-03
Example IV	Best	3.124e-06	1.768e-08	8.645e-06	1.139e-05	1.277e-03	1.316e-05
	Worst	3.054e-01	5.904e-06	2.456e+00	8.225e-03	5.007e-02	4.019e-03
	Average	8.206e-03	2.540e-06	2.037e-01	1.449e-03	1.220e-02	6.618e-04
	Median	3.056e-04	2.715e-06	1.151e-03	9.029e-04	1.091e-02	2.033e-04
	SD	4.308e-02	1.280e-06	5.263e-01	1.760e-03	9.180e-03	9.653e-04
Example V	Best	5.336e-04	2.964e-31	5.072e-04	7.645e-04	1.811e-03	2.416e-05
	Worst	3.620e-02	3.139e-09	2.673e-01	3.050e-02	1.015e-01	1.191e-02
	Average	8.303e-03	6.453e-11	1.790e-02	9.688e-03	1.746e-02	2.697e-03
	Median	9.395e-03	2.014e-18	8.639e-03	9.184e-03	1.518e-02	8.339e-04
	SD	6.501e-03	4.437e-10	4.837e-02	5.463e-03	1.471e-02	3.745e-03

Table 19
The parameter estimation of algorithms for Example I.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	0.050	0.050	0.050	0.050	0.049	0.043	0.050
b_1	−0.400	−0.400	−0.400	−0.400	−0.399	−0.393	−0.400
a_1	1.131	1.131	1.131	1.131	1.132	1.133	1.131
a_2	−0.250	−0.250	−0.250	−0.250	−0.251	−0.2514	−0.250

Table 20
The parameter estimation of algorithms for Example II.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	−0.20	−0.20	−0.20	−0.20	−0.19	−0.19	−0.20
b_1	−0.40	−0.40	−0.40	−0.40	−0.39	−0.39	−0.40
b_2	0.50	0.50	0.50	0.50	0.51	0.51	0.50
a_1	0.60	0.60	0.60	0.60	0.61	0.64	0.60
a_2	−0.25	−0.25	−0.25	−0.25	−0.24	−0.23	−0.25
a_3	0.20	0.20	0.20	0.20	0.21	0.21	0.20

Table 21
The parameter estimation of algorithms for Example III.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	1.0000	1.0000	1.0000	0.9998	0.9996	1.0250	0.9991
b_1	−0.9000	−0.9000	−0.9000	−0.9042	−0.8928	−0.9313	−0.8975
b_2	0.8100	0.8100	0.8100	0.8093	0.8071	0.7354	0.8154
b_3	−0.7290	−0.7290	−0.7290	−0.7309	−0.7283	−0.7019	−0.7261
a_1	−0.0400	−0.0400	−0.0400	−0.0362	−0.0463	−0.0031	−0.0424
a_2	−0.2775	−0.2775	−0.2775	−0.2733	−0.2795	−0.1994	−0.2855
a_3	0.2101	0.2101	0.2101	0.2131	0.2110	0.2579	0.2018
a_4	−0.1400	−0.1400	−0.1400	−0.1395	−0.1363	−0.1390	−0.1442

Table 22
The parameter estimation of algorithms for Example IV.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	0.1084	0.1083	0.1084	0.1083	0.1106	0.1040	0.1078
b_1	0.5419	0.4900	0.4609	0.4536	0.4676	0.4798	0.4332
b_2	1.0840	0.8416	0.7088	0.6748	0.7262	0.7193	0.6129
b_3	1.0840	0.6541	0.4176	0.3589	0.3822	0.4713	0.3065
b_4	0.5419	0.1908	−0.0023	−0.0484	−0.1161	0.03742	−0.0184
b_5	0.1084	−0.0038	−0.0675	−0.0821	−0.1463	−0.0045	−0.0189
a_1	−0.9853	−0.5027	−0.2384	−0.1682	−0.3329	−0.3064	−0.0012
a_2	−0.9738	−0.6778	−0.5157	−0.4804	−0.3086	−0.5344	−0.5806
a_3	−0.3864	−0.0534	0.1316	0.1806	0.1153	−0.0100	0.2203
a_4	−0.1112	−0.0536	−0.0227	−0.0183	0.0952	−0.0090	−0.0916
a_5	−0.0113	0.0079	0.0192	0.0236	−0.0006	−0.0013	0.0267

Table 23
The parameter estimation of algorithms for Example V.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	1.0000	0.9952	1.0000	0.9911	1.0030	0.9290	1.0020
b_2	−0.4000	0.0951	−0.4016	0.0310	0.8110	0.3474	0.1533
b_4	−0.6500	−0.4787	−0.6504	−0.4870	−0.2160	−0.2686	−0.4490
b_6	0.2600	0.0587	0.2606	0.0794	−0.2203	−0.0571	0.0338
a_2	0.7700	0.2884	0.7716	0.3439	−0.4426	−0.0001	0.2150
a_4	0.8498	0.8518	0.8497	0.8420	0.8445	0.8281	0.8424
a_6	−0.6486	−0.2361	−0.6499	−0.2729	0.4056	0.0442	−0.1625

Table 24
Experimental results (MSE) of algorithms for Test Examples VI, VII, VIII, IX, X.

Problems	Measure	Mean square error (MSE)					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
Example VI	Best	4.779e-22	0	6.777e-27	9.614e-07	5.878e-05	0
	Worst	2.158e-20	0	3.560e+00	2.908e+00	5.688e-01	0
	Average	6.638e-21	0	7.121e-02	8.731e-02	1.494e-02	0
	Median	5.104e-21	0	1.456e-18	6.153e-06	2.239e-03	0
	SD	5.239e-21	0	5.034e-01	4.234e-01	8.003e-02	0
Example VII	Best	6.540e-23	0	0	1.156e-07	6.273e-05	0
	Worst	3.137e-21	2.128e-31	0	5.108e-06	3.023e-03	0
	Average	1.157e-21	8.506e-33	0	1.523e-06	7.333e-04	0
	Median	9.449e-22	0	0	1.194e-06	4.279e-04	0
	SD	7.713e-22	4.209e-32	0	1.142e-06	6.875e-04	0
Example VIII	Best	1.626e-22	0	0	3.298e-08	1.997e-05	0
	Worst	2.581e-21	4.578e-33	0	7.673e-03	8.759e-03	0
	Average	8.244e-22	9.156e-35	0	9.694e-04	1.005e-03	0
	Median	6.897e-22	0	0	2.495e-07	1.737e-04	0
	SD	5.130e-22	6.475e-34	0	2.432e-03	2.289e-03	0
Example IX	Best	9.748e-14	0	1.798e-05	2.513e-05	7.443e-04	0
	Worst	8.902e-02	0	2.059e-01	3.162e+00	1.448e-01	0
	Average	6.291e-03	0	4.343e-02	1.076e-01	4.313e-02	0
	Median	6.838e-04	0	3.058e-02	2.315e-02	2.312e-02	0
	SD	1.461e-02	0	4.427e-02	4.426e-01	4.259e-02	0
Example X	Best	1.772e-21	0	2.172e-16	9.734e-07	2.365e-04	0
	Worst	9.592e-02	2.640e-32	8.775e-02	8.765e-02	9.275e-02	0
	Average	5.284e-03	2.833e-33	7.671e-03	1.015e-02	1.519e-02	0
	Median	1.582e-17	0	1.450e-09	2.376e-03	5.322e-03	0
	SD	2.116e-02	7.816e-33	2.154e-02	1.775e-02	2.393e-02	0

Table 25
The parameter estimation of algorithms for Example VI.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	1.00	1.00	1.00	1.00	1.01	1.01	1.00
a_1	1.40	1.40	1.40	1.40	1.40	1.39	1.40
a_2	−0.49	-0.49	−0.49	-0.49	-0.48	-0.48	−0.49

Table 26
The parameter estimation of algorithms for Example VII.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	1.00	1.00	1.00	1.00	0.99	0.99	1.00
a_1	1.20	1.20	1.20	1.20	1.20	1.201	1.20
a_2	−0.60	−0.60	−0.60	−0.60	−0.60	−0.61	−0.60

Table 27
The parameter estimation of algorithms for Example VIII.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_1	1.25	1.25	1.25	1.25	1.25	1.24	1.25
b_2	−0.25	−0.25	−0.25	−0.25	−0.25	−0.24	−0.25
a_1	0.30	0.30	0.30	0.30	0.30	0.29	0.30
a_2	−0.40	−0.40	−0.40	−0.40	−0.40	−s0.40	−0.40

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} +, \cdots, + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} +, \cdots, + a_m z^{-m}} \tag{5}$$

where $Y(z)$and $U(z)$are the output and input, respectively, $\boldsymbol{a} = [a_1, a_2, \ldots, a_n]$and $\boldsymbol{b} = [b_0, b_1, \ldots, b_n]$ are the real coefficients of the numerator and denominator, $n$and $m$are the maximum degree of the polynomials of the numerator and denominator respectively.

According to Fig. 11, the problem of the adaptive IIR filter is to identify the unknown model whose transfer function is described by Eq. 5, $\boldsymbol{d}_k$ is the output of the unknown system in time k, and a noise component $\boldsymbol{v}_k$is added to generate the output $\boldsymbol{y}_k$. The adaptive IIR filter block receives the same input as the unknown IIR system block $\boldsymbol{u}_k$. It also receives as input the coefficients of the numerator and denominator generated by the metaheuristic algorithm $\Theta_k = [\boldsymbol{a}, \boldsymbol{b}]$, the output generated is an estimated response $\widehat{\boldsymbol{y}}_k$. The deviation of the estimated signal $\widehat{\boldsymbol{y}}_k$to the output signal of the unknown system $\boldsymbol{y}_k$ is known as the mean square error (MSE) and is represented by e_k. Mathematically, the adaptive IIR filter problem can be modeled as an optimization problem of minimizing the signal e_k, that is:

$$min\ e_k = \frac{1}{L}\sum_{i=1}^{L}(y(i) - \widehat{y}(i))^2 \tag{6}$$

where L is the length of the vector or input signal $\boldsymbol{u}_k$.

The 10 cases analyzed in this work are the IIR filter models reviewed in the references (Krusienski & Jenkins, 2005; Panda, Pradhan, & Majhi, 2011; Jiang et al., 2015; Lagos-Eulogio et al., 2017; Zhao et al., 2019). Table 17 shows the models for Examples I, II, III, IV, V, VI, VII, VIII, IX, and X. The identification of parameters is carried out through a transfer function of full order.

It is necessary to mention that the adaptive IIR filter design problem requires robust algorithms since the systems are non-linear, non-differentiable, and multimodal. Recent hybrid algorithms have been proposed for this problem. For example, in Jiang et al. (2015), a hybrid algorithm is implemented with a particle swarm optimization (PSO) and a gravitational search algorithm (GSA), which is called the HPSO-GSA. This algorithm generates a co-evolutionary technique, taking advantage of PSO's ability to share information between particles and the memory of the best position obtained, and the GSA's force and mass acceleration to determine the best search direction. Similarly, in Lagos-Eulogio et al. (2017) the parameter estimation problem is addressed using a hybrid algorithm called CPSO-DE, which is a combination of concepts of a cellular-automata neighborhood, PSO, and differential evolution (DE). In this way, the CPSO-DE is able to modify the trajectory of the particles to leave the local optimum, while the PSO and the DE focus on finding a higher quality solution. The performance of the CCAA is compared with two algorithms that also use cellular automata concepts (CPSO and CPSO-DE), with a gravitational hybrid algorithm (HPSO-GSA), and with two recent algorithms (GWO and MSCA). To make a more fair comparison, a population of 100individuals was applied for the HPSO-GSA, GWO, and MSCA. For the CPSO, CPSO-DE and CCAA, 20 smart-cells and 5 neighbors for smart-cell were utilized. Besides, 50 runs were carried

Table 28
The parameter estimation of algorithms for Example IX.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	1.000	0.999	1.000	0.999	0.995	0.969	1.000
a_1	1.500	1.500	1.500	1.501	1.507	1.524	1.500
a_2	−0.750	−0.750	−0.750	−0.751	−0.761	−0.786	−0.750
a_3	0.125	0.125	0.125	0.125	0.130	0.140	0.125

Table 29
The parameter estimation of algorithms for Example X.

Parameters	Actual values	Parameter estimation value					
		HPSO-GSA	CPSO-DE	CPSO	GWO	MSCA	CCAA
b_0	−0.300	−0.300	−0.300	−0.300	−0.300	−0.328	−0.300
b_1	0.400	0.400	0.400	0.400	0.400	0.422	0.400
b_2	−0.500	−0.500	−0.500	−0.500	−0.500	−0.522	−0.500
a_1	1.200	1.200	1.200	1.200	1.194	1.169	1.200
a_2	−0.500	−0.500	−0.500	−0.500	−0.487	−0.471	−0.500
a_3	0.100	0.100	0.100	0.100	0.093	0.096	0.100

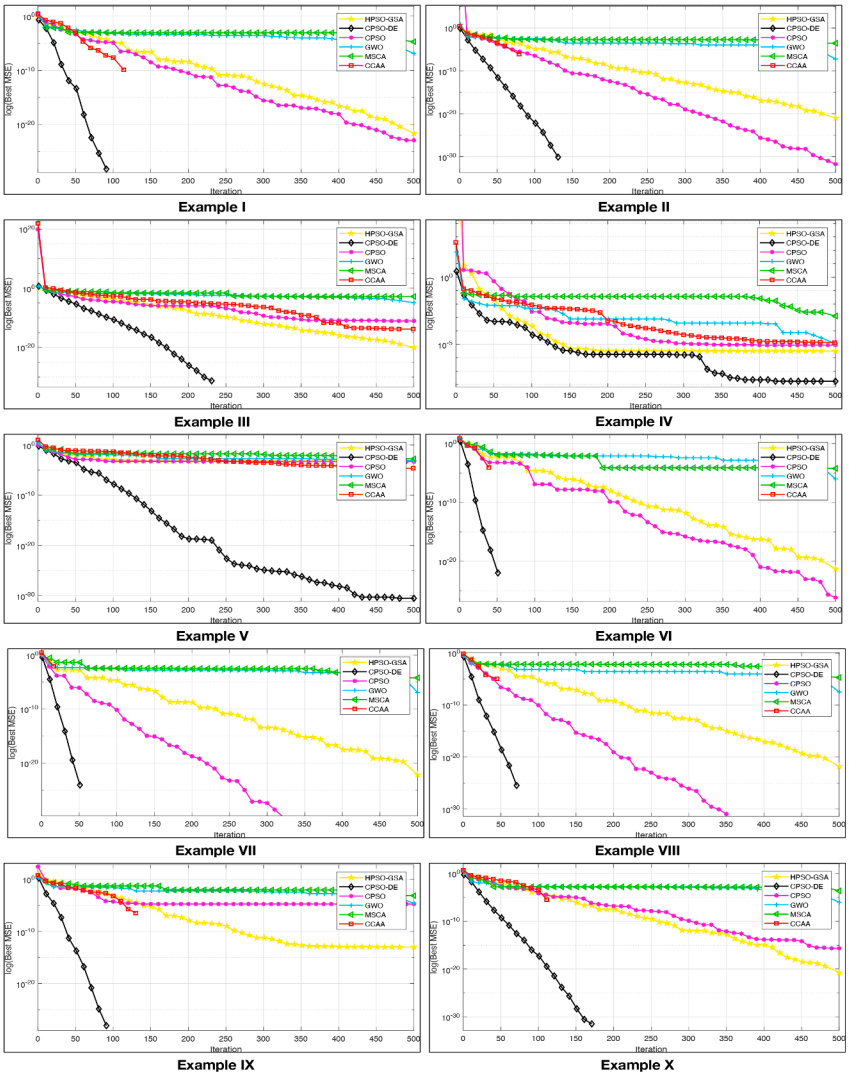

Fig. 12. Convergence of the MSE curve for the IIR filters.

out for every algorithm, each applying a total number of 500 iterations.

Table 18 shows the MSE statistical results for the best, worst, average, mean, and standard deviation for each example for the first 5 IIR filters. For Cases I and II, the CCAA and the CPSO-DE algorithm identified the parameters with an MSE of 0, obtaining the best of the 50 runs. For Cases III to V, the CPSO-DE algorithm obtained a higher performance than the others for calculating the best parameter estimation. In these cases, the CCAA obtained positions 3, 3 and 2, respectively, in the best value, and positions 3, 2 and 2 in the best average value. The best parameter estimations for these 5filters are presented in Tables 19–23.

For Cases VI, VII, VIII, IX, and X, the CCAA achieves better performance than the other algorithms (Table 24). In all cases, the CCAA achieves an MSE of 0 for all statistics, the CPSO algorithm achieves it for Cases VII and VIII, and CPSO-DE for Case IX.

The best parameter estimations for the last 5 filters are presented in Tables 25–29, and the convergence curves in the parameter estimation of the 10 cases are presented in Fig. 12.

In summary, for Cases I, II, and VI to X, the CCAA achieved complete parameter identification, and for Cases I and VI to X, the CCAA obtained an MSE of 0 in all the statistics. In general, for the 10 cases, the CCAA, CPSO-DE and CPSO were the only algorithms with complete parameter identification: the CPSO-DE in 8 cases and the CCAA in 7 cases. The CCAA had 6 best average values, and the CPSO-DE had 7 best average values, resulting in the best algorithms overall.

These results demonstrate that the CCAA can be very useful for the design of adaptive IIR filters, and that the metaheuristics applying cellular automata concepts can be extremely helpful to optimize challenging problems. Note that the source codes of the CCAA can be downloaded from https://github.com/juanseck/CCAA.git.

7. Conclusions and future work

This study presented a new global optimization algorithm called the CCAA, which uses a neighborhood and the evolution rules inspired by cellular automata concepts, such as local interactions between solutions and different evolution rules. These elements allow the exploration and exploitation operations to be carried out concurrently and randomly for each smart-cell in the solution population.

This strategy allows the CCAA to establish a proper balance of local and global changes and information exchange between smart-cells during the search for the optimal solution. The experimental fine-tune of parameters and the algorithm validation was carried out with 48test functions that were widely used to analyze the exploration, the

exploitation, the escape of local optima, and the convergence behavior of the proposed algorithm, analyzing the performance of the CCAA for problems on 30, 500, and fixed dimensions.

The experimental results obtained were compared with 14 of the most representative and recent computational intelligence algorithms for global optimization recognized for their excellent performance; namely, ADIFA, CPSO, EHO, GWO, HHO, LSHADE, LSPACMA, MA, MBO, MSA, MSCA, RDWOA, SMA, and WOA. The comparison showed that the CCAA obtained satisfactory results and is competitive enough among the other algorithms.

To verify the performance of the CCAA, 5 engineering design problems (GTD, PVD, WBD, CBD, and IIR filter design) were also addressed. The results also demonstrated the effectiveness of the CCAA to find quality solutions in these types of problems, showing that the CCAA is very competitive with respect to recently published metaheuristic optimizers.

In future applications, the intention is to apply improvements of the CCAA for other engineering problems, such as the design of reduced-order IIR filters. This article proposed the application of the CCAA to solve single-objective optimization problems. However, many problems involve multiple objectives for real situations. In this sense, a proposal for future work is to use the CCCAA to optimize continuous and discrete multi-objective problems, such as task scheduling problems, route scheduling, aeronautical modeling problems, classification problems, and power dispatch problems, among others. The application of different and diverse evolution rules is also proposed to improve the performance of the CCAA.

CRediT authorship contribution statement

Juan Carlos Seck-Tuoh-Mora: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft. **Norberto Hernandez-Romero:** Conceptualization, Methodology, Validation, Writing - original draft. **Pedro Lagos-Eulogio:** Conceptualization, Software, Validation, Formal analysis, Writing - original draft. **Joselito Medina-Marin:** Methodology, Investigation, Writing - original draft. **Nadia Samantha Zuñiga-Peña:** Investigation, Methodology, Writing - original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This study was supported by the National Council for Science and Technology (CONACYT) with project number CB- 2017-2018-A1-S-43008. Nadia S. Zuñiga-Peña was supported by CONACYT Grant No. 785376.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.eswa.2021.114930.

References

Atul Kumar Dwivedi, S. G. & Londhe, N. D. (2018). Review and analysis of evolutionary optimization-based techniques for fir filter design. Circuits Systems Signal Processing 37, 4409–4430.

Bilan, S. M., Bilan, M. M., & Motornyuk, R. L. (2020). New methods and paradigms for modeling dynamic processes based on cellular automata. *IGI Global*.

Chen, H., Yang, C., Heidari, A. A., & Zhao, X. (2020). An efficient double adaptive random spare reinforced whale optimization algorithm. *Expert Systems with Applications, 154*, Article 113018.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*, 113–127.

Cui, Z., & Gao, X. (2012). Theory and applications of swarm intelligence. *Neural Computing and Applications, 2*, 205–206.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine, 1*, 28–39.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*, 53–66.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. In MHS'95. Proceedings of the sixth international symposium on micro machine and human science (pp. 39–43). IEEE.

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17*, 4831–4845.

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*, 17–35.

Gupta, S., Deep, K., Mirjalili, S., & Kim, J. H. (2020). A modified sine cosine algorithm with novel transition parameter and mutation operator for global optimization. In *Expert Systems with Applications* (p. 113395).

Harris, F. J. (2004). *Multirate signal processing for communication systems*. Prentice Hall PTR.

Hassanien, A. E., & Emary, E. (2018). *Swarm intelligence: Principles, advances, and applications*. CRC Press.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97*, 849–872.

Hernández-Gress, E. S., Seck-Tuoh-Mora, J. C., Hernández-Romero, N., Medina-Marín, J., Lagos-Eulogio, P., & Ortíz-Perea, J. (2020). The solution of the concurrent layout scheduling problem in the job-shop environment through a local neighborhood search algorithm. *Expert Systems with Applications, 144*, Article 113096.

Hoekstra, A. G., Kroc, J., & Sloot, P. M. (2010). *Simulating complex systems by cellular automata*. Springer.

Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*, 150–194.

Jiang, S., Wang, Y., & Ji, Z. (2015). A new design method for adaptive iir system identification using hybrid particle swarm optimization and gravitational search algorithm. *Nonlinear Dynamics, 79*, 2553–2576.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

Krusienski, D. J., & Jenkins, W. K. (2005). Design and performance of adaptive systems based on structured stochastic optimization strategies. *IEEE Circuits and Systems Magazine, 5*, 8–20.

Kumar, K., & Davim, J. P. (2020). *Optimization using evolutionary algorithms and metaheuristics: Applications in engineering* (1st Ed.). CRC Press.

Lagos-Eulogio, P., Seck-Tuoh-Mora, J., Hernandez-Romero, N., & Medina-Marin, J. (2017). A new design method for adaptive iir system identification using hybrid cpso and de. *Nonlinear Dynamics, 88*, 2371–2389.

Li, S., Chen, H., Wang, M., Heidari, A. A. & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems.

McIntosh, H. V. (2009). *One dimensional cellular automata*. Luniver Press.

de Melo, V. V., & Banzhaf, W. (2018). Drone squadron optimization: A novel self-adaptive algorithm for global numerical optimization. *Neural Computing and Applications, 30*, 3117–3144.

Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

Mohamed, A. W., Hadi, A. A., Fattouh, A. M., & Jambi, K. M. (2017). Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems. In *2017 IEEE Congress on evolutionary computation (CEC)* (pp. 145–152). IEEE.

von Neumann, J. (1966). *Theory of self-reproducing automata*. Champain, IL: University of Illionois Press.

Panda, G., Pradhan, P. M., & Majhi, B. (2011). Iir system identification using cat swarm optimization. *Expert Systems with Applications, 38*, 12671–12683.

Salcedo-Sanz, S. (2016). Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. *Physics Reports, 655*, 1–70.

Salcido, A. (2011). *Cellular automata: Innovative modelling for science and engineering*. BoD–Books on Demand.

Salih, S. Q., Alsewari, A. A., & Yaseen, Z. M. (2019). Pressure vessel design simulation: Implementing of multi-swarm particle swarm optimization. In *Proceedings of the 2019 8th international conference on software and computer applications* (pp. 120–124). New York, NY, USA: Association for Computing Machinery.

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design, 112*, 223–229.

Sarker, R., Kamruzzaman, J., & Newton, C. (2003). Evolutionary optimization (evopt): A brief review and analysis. *International Journal of Computational Intelligence and Applications, 3*, 311–330.

Schiff, J. L. (2011). *Cellular automata: A discrete view of the world* (Vol. 45). John Wiley & Sons.

Seredynski, F., & Zomaya, A. Y. (2002). Sequential and parallel cellular automata-based scheduling algorithms. *IEEE Transactions on Parallel and Distributed Systems, 13*, 1009–1023.

Shi, Y., Liu, H., Gao, L., & Zhang, G. (2011). Cellular particle swarm optimization. *Information Sciences, 181*, 4460–4493.

Tanabe, R., & Fukunaga, A. S. (2014). Improving the search performance of shade using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)* (pp. 1658–1665). IEEE.

Dokeroglua, Tansel, Ender Sevincb, T. K., & Cosarb, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering, 137*, 4409–4430.

Van Den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences, 176*, 937–971.

Wang, G. G. (2018). Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Computing, 10*, 151–164.

Wang, G. G., Deb, S. & Coelho, L. d. S. (2015). Elephant herding optimization. In 2015 3rd International symposium on computational and business intelligence (ISCBI) (pp. 1–5). IEEE.

Wang, G. G., Deb, S., & Coelho, L. D. S. (2018). Earthworm optimisation algorithm: A bio-inspired metaheuristic algorithm for global optimisation problems. *International Journal of Bio-Inspired Computation, 12*, 1–22.

Wang, G. G., Deb, S., & Cui, Z. (2019). Monarch butterfly optimization. *Neural Computing and Applications, 31*, 1995–2014.

Wang, G. G., Gandomi, A. H., & Alavi, A. H. (2014). Stud krill herd algorithm. *Neurocomputing, 128*, 363–370.

Wang, G. G., Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2014). A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. *Engineering Computations*.

Wolfram, S. (2002). *A new kind of science* (Vol. 5). IL: Wolfram media Champaign.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*, 67–82.

Wu, J., Wang, Y. G., Burrage, K., Tian, Y. C., Lawson, B., & Ding, Z. (2020). An improved firefly algorithm for global continuous optimization problems. *Expert Systems with Applications, 113340*.

Zervoudakis, K., & Tsafarakis, S. (2020). A mayfly optimization algorithm. *Computers & Industrial Engineering, 106559*.

Zhao, R., Wang, Y., Liu, C., Hu, P., Jelodar, H., Yuan, C., Li, Y., Masood, I., Rabbani, M., Li, H., et al. (2019). Selfish herd optimization algorithm based on chaotic strategy for adaptive iir system identification problem. *Soft Computing*, 1–48.