Journal homepage http://revistas.unitru.edu.pe/index.php/SSMM

SELECCIONES MATEMÁTICAS
Universidad Nacional de Trujillo
ISSN: 2411-1783 (Online)
2019; Vol.06(2): 156 - 177.

Enhancement of the Multi-Particle Collision Algorithm by mechanisms derived from the Opposition-Based Optimization.

Mejora del Algoritmo de Colisión de Múltiples Partículas utilizando mecanismos derivados de optimización basada en oposición.

Reynier Hernández Torres*, Haroldo F. Campos Velho †, and Eduardo F. P. da Luz‡

Received, Oct. 31, 2019 *Accepted, Nov. 30, 2019*

How to cite this article:
Hernandez, R., et al. *Enhancement of the Multi-Particle Collision Algorithm by mechanisms derived from the Opposition-Based Optimization*. Selecciones Matemáticas. 2019; 6(2):156-177. http://dx.doi.org/10.17268/sel.mat.2019.02.03

Abstract

New versions of the metaheuristic Multi-Particle Collision Algorithm (MPCA) are presented. In order to provide more effective candidate solutions for an optimization problem, the concept of opposition and reflection is introduced to improve the capacity to find a solution in the search space. Four different strategies to compute the reflected and opposite points are implemented. The performance of all implementations is evaluated over thirty objective functions with different complexities, using serial and parallel versions of the algorithms.

Keywords. Stochastic algorithm; Metaheuristic; Opposition-Based Learning; Reflection-Based Learning; Multi-Particle Collision Algorithm.

Resumen

En este trabajo se presentan nuevas versiones de la metaheurística Algoritmo de Colisión de Múltiples Partículas (MPCA). Para proporcionar soluciones candidatas más efectivas para un problema de optimización, se introduce el concepto de oposición y reflexión, con el objetivo de mejorar la capacidad de encontrar una solución en el espacio de búsqueda. Se implementan cuatro estrategias diferentes para calcular los puntos reflejados y opuestos. El rendimiento de todas las implementaciones se evalúa en más de treinta funciones objetivo con diferentes complejidades, utilizando versiones en serie y paralelas de los algoritmos.

Palabras clave. Algoritmo estocástico; Metaheurística; Aprendizaje basado en oposición; Aprendizaje basado en reflexión; Algoritmo de Colisión de Múltiples partículas.

1. Introduction. Optimization is an area of the applied mathematics covering theory and techniques used to minimize or maximize a determined objective function (also called cost function or error function), within a defined domain (or set of constraints) [6, 73].

With recent advances in the computer science area, many techniques have been developed in the sub-area of stochastic optimization methods. Those algorithms are called to improve the exploration of the search space, making it more efficient, expectedly converging more quickly to the global optimum.

Metaheuristic algorithms are powerful tools that can solve optimization problems with a high number of variables [73]. These problems usually could not be solved by other deterministic optimization algorithms in a reasonable time [42].

*Associated Laboratory for Computing and Applied Mathematics (LAC), National Institute for Space Research (INPE), São José dos Campos, São Paulo, Brazil (reynier.torres@inpe.br)

†Associated Laboratory for Computing and Applied Mathematics (LAC), National Institute for Space Research (INPE), São José dos Campos, São Paulo, Brazil (haroldo.camposvelho@inpe.br)

‡Centro Nacional de Monitoramento e Alertas de Desastres Naturais (Cemaden), São José dos Campos, São Paulo, Brazil (eduardo.luz@cemaden.gov.br)

A huge number of metaheuristics can be found in the literature [16, 31]. Many algorithms are usually inspired by evolution, such as the most known Evolutionary Programming (EP) [20], Evolution Strategies (ESs) [7], Genetic Algorithms (GAs) [29], Genetic Programming (GP) [39], Differential Evolution (DE) [11], Cultural Algorithms (CA) [56], and Biogeography-Based Optimization (BBO) [62]. Another important number of metaheuristics are based on swarm intelligence: Particle Swarm Optimization (PSO) [17, 36], Ant Colony Optimization (ACO) [14, 15], Artificial Bee Optimization (ABC) [35], Bacterial foraging optimization (BFO) [10], Intelligent Water Drops (IWS) [60], and Artificial Immune Systems (AIS) [12]. Other metaheuristics are human-based, such as Memetic Algorithms (MA) [48], and Harmony search (HS) [25], based on the behavior of musicians. Another class of metaheuristics is classified as sciences-based. Some algorithms belonging this group are Simulated Annealing (SA) [37], Particle Collision Algorithm (PCA) [57] and Multi-Particle Collision Algorithm (MPCA) [43].

MPCA [43] is a metaheuristic algorithm based on the canonical Particle Collision Algorithm (PCA) [57, 58], where particles (candidate solution) travel in the search space and cooperate among themselves, sharing the best particle solution found after some objective function evaluations. MPCA has been successfully used in the solution of many optimization problems, such as diagnostic of hydromechanical systems [64], fault diagnosis [18], automatic configuration for neural network applied to different problems such as atmospheric temperature profile identification [59], data assimilation [5], and climate prediction [4].

The concept of Opposition-based learning (OBL) was proposed by Tizhoosh [65]. The idea of this optimization technique is to find a better solution searching in the opposite area of a particular candidate solution.

This work introduces new variants of the MPCA that exploits the advantages of parallel computation, and the opposition and reflection concepts. Hence OBL is used to accelerate MPCA. Four variants will be compared: Opposite MPCA (OMPCA), Quasi-Opposite MPCA (QOMPCA), Quasi-Reflective MPCA (QRMPCA) and Center-Based Sampling MPCA (CBMPCA).

The paper is organized as follows: the section 2 introduces MPCA; the section 3 reviews the opposition concepts; the section 4 describes the proposed variants (OMPCA, QOMPCA, QRMPCA, CBMPCA); the section 5 presents the experimental results, and the section 6 concludes the work.

2. Multi-Particle Collision Algorithm (MPCA). The PCA was inspired by the physics of nuclear particle collision reactions [57, 58]. In the nuclear reactor, some phenomena occur, including scattering (an incident particle is scattered by a target nucleus), and absorption (an incident particle is absorbed by the target nucleus), as shown in Figure 2.1. A *Particle* is a candidate solution. If a new particle has a better fitness, it replaces the old one, meaning an *absorption*. If a new particle has a worse fitness than a previous one, there will be a probability of looking for another particle in a region far from the local place of the old particle in the search space, emulating a *scattering*.

MPCA can be loosely described as an algorithm consisting of a set of particles traveling inside a nuclear reactor. New particles are generated, and they can be absorbed or scattered, depending on their fitness, and, if the fitness is better, they will substitute the *old* particles.

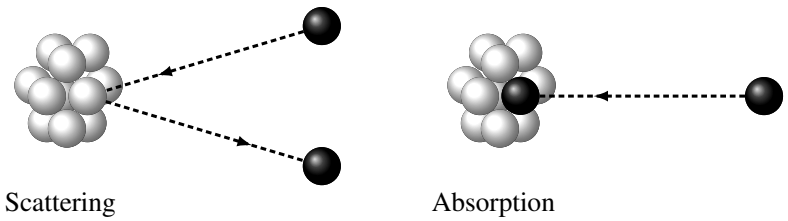


FIGURE 2.1. *Phenomena inspiring Multi-Particle Collision Algorithm*

A parallel version of the MPCA was developed, taking advantages of a high-performance environment, and its pseudo-code is shown in Algorithm 1. In each processor (of a total of $N_{\text{processors}}$), $N_{\text{particles}}$ candidate solutions are set. This partition leads to a considerable reduction of computing time [43].

MPCA starts with initial particles randomly created spread all across the search space (Algorithm 1 – lines 3 and 4). After creating the initial set, a blackboard strategy is used for sharing the best particle among all the particles (Algorithm 1 – lines 6 and 7). Later, the particles traveling process is started, involving three main functions: Perturbation (see subsection 2.1), Exploitation (see Subsection subsection 2.2) and Scattering (see subsection 2.3) [43, 57].

The PERTURBATION function represents the particles displacement in the nuclear reactor, while the EXPLOITATION function emulates the absorption phenomenon, and the SCATTERING corresponds to the scattering phenomenon in the reactions.

After each iteration, if a number of function evaluations ($N_{FE_{\text{blackboard}}}$) was reached after the last blackboard updating (found as N_{FE_i} - $N_{FE_{\text{lastUpdate}}}$ in Algorithm 1 – line 19), then the mechanism of cooperation is triggered (Algorithm 1 – lines 20-22). Again, the best particle is shared among all the particles in the set.

Algorithm 1 Multi-Particle Collision Algorithm

1: Set MPCA control parameters ($N_{\text{processors}}$: number of processors; $N_{\text{particles}}$: number of particles in each processor; $N_{FE_{\text{mpca}}}$: maximum number of function evaluations; $N_{FE_{\text{blackboard}}}$: trigger for update blackboard; LB, UB: lower and upper bounds in solution vector, respectively; IL, SL: inferior and superior limits for perturbation, respectively)
2: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do** ▷ Initial set of particles
3: $N_{FE_i} = 0$, $N_{FE_{\text{lastUpdate}}} = 0$
4: **for** $j \leftarrow 1$ **to** $N_{\text{particles}}$ **do**
5: **currentP**$_{i,j}$ = RANDOMSOLUTION
6: $N_{FE_i} = N_{FE_i} + 1$
7: **end for**
8: **bestP**$_i$ = UPDATEBLACKBOARD ▷ Initial blackboard
9: **end for**
10: **while** $NFE_{\text{total}} < N_{FE_{\text{mpca}}}$ **do** ▷ Stopping criteria
11: $NFE_{\text{total}} = 0$
12: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do**
13: **for** $j \leftarrow 1$ **to** $N_{\text{particles}}$ **do**
14: **newP**$_{i,j}$ = PERTURBATION(**currentP**$_{i,j}$)
15: **if** f(**newP**$_{i,j}$) $< f$(**currentP**$_{i,j}$) **then**
16: **currentP**$_{i,j}$ = **newP**$_{i,j}$
17: **currentP**$_{i,j}$ = EXPLORATION(**currentP**$_{i,j}$)
18: **else**
19: **currentP**$_{i,j}$ = SCATTERING(**currentP**$_{i,j}$, **newP**$_{i,j}$, **bestP**$_i$)
20: **end if**
21: **if** f(**currentP**$_{i,j}$) $< f$(**bestP**$_i$) **then**
22: **bestP**$_i$ = **currentP**$_{i,j}$
23: **end if**
24: **end for**
25: **if** N_{FE_i} - $N_{FE_{\text{lastUpdate}}} > N_{FE_{\text{blackboard}}}$ **then**
26: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do** ▷ Blackboard
27: **bestP**$_i$ = UPDATEBLACKBOARD
28: $N_{FE_{\text{lastUpdate}}} = N_{FE_i}$
29: **end for**
30: **end if**
31: $NFE_{\text{total}} = NFE_{\text{total}} + N_{FE_i}$
32: **end for**
33: **end while**
34: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do** ▷ Final blackboard
35: **bestP**$_i$ = UPDATEBLACKBOARD
36: **end for**
37: **return bestP**$_1$

As stopping criterion, a maximum number of function evaluations ($N_{FE_{\text{mpca}}}$) is defined.

The current version MPCA was implemented in C++, using MPI libraries, in a multiprocessor architecture with distributed memory machine. OpenMPI libraries were used for parallel processing.

2.1. Perturbation function. The PERTURBATION function (see Algorithm 2 – ignore line 10) performs a random variation of a particle within a defined range. The new perturbated particle **P** is found by computing each dimension d as:

$$\mathbf{P}_{(d)} = \mathbf{P}^{\star}_{(d)} + ((\mathbf{UB}_{(d)} - \mathbf{P}^{\star}_{(d)}) \cdot \mathrm{R}) - ((\mathbf{P}_{(d)} - \mathbf{LB}_{(d)}) \cdot (1 - \mathrm{R})), \tag{2.1}$$

where $\mathbf{P}^{\star}$ is the particle to be perturbed, **UB** and **LB** are the upper and the lower limits over the defined search space, respectively, and R is a random number uniformly generated between 0 and 1.

Algorithm 2 Perturbation Function

1: **function** PERTURBATION(currentP)
2: **for** $d \leftarrow 1$ **to** D **do**
3: R = $rand(0, 1)$
4: $\mathbf{P}_{(d)} = \mathbf{P}^{\star}_{(d)} + ((\mathbf{UB}_{(d)} - \mathbf{P}^{\star}_{(d)}) \cdot \mathrm{R}) - ((\mathbf{P}_{(d)} - \mathbf{LB}_{(d)}) \cdot (1 - \mathrm{R}))$
5: **if** $\mathbf{P}_{(d)} > \mathbf{UB}_{(d)}$ **then**
6: $\mathbf{P}_{(d)} = \mathbf{UB}_{(d)}$
7: **else if** $\mathbf{P}_{(d)} < \mathbf{LB}_{(d)}$ **then**
8: $\mathbf{P}_{(d)} = \mathbf{LB}_{(d)}$
9: **end if**
10: **end for**
11: $NFE = NFE + 1$
12: **P** = OPPOSITION(**P**)
13: **return** P
14: **end function**

2.2. Exploitation function. If the new perturbed particle **P** is better that the current particle **currentP**, then the EXPLOITATION function (see Algorithm 3 – ignore lines 5 and 6) performs an intensification of the particle. In this intensification stage, a series of small perturbations are performed, computing a new particle **newP** each time (see Algorithm 4), using the following equation:

$$\textbf{newP}_{(d)} = \textbf{currentP}_{(d)} + ((u - \textbf{currentP}_{(d)}) \cdot \mathrm{R}) - ((\textbf{currentP}_{(d)} - l) \cdot (1 - \mathrm{R})), \tag{2.2}$$

where u and l are the new upper and lower limits computed from the current particle **currentP**, using the superior and inferior values IL and SL for the limits of the random number generated.

Algorithm 3 Exploitation Function

```
1: function EXPLOITATION(currentP)
2:     for n ← 1 to N_{FE_internalMpca} do
3:         newP = SMALLPERTURBATION(currentP)
4:         NFE = NFE + 1
5:         if rand(0,1) < J_r then
6:             newP = OPPOSITION(newP)
7:         end if
8:         if f(newP) < f(currentP) then
9:             currentP = newP
10:        end if
11:    end for
12:    return currentP
13: end function
```

Algorithm 4 Small Perturbation Function

```
1: function SMALLPERTURBATION(currentP)
2:     for d ← 1 to D do
3:         u = currentP_(d) · rand(1, SL)
4:         l = currentP_(d) · rand(IL, 1)
5:         R = rand(0,1)
6:         if u > UB_(d) then
7:             u = UB_(d)
8:         end if
9:         if l < LB_(d) then
10:            l = LB_(d)
11:        end if
12:        newP_(d) = currentP_(d) + ((u − currentP_(d)) · R) − ((currentP_(d) − l) · (1 − R))
13:    end for
14:    NFE = NFE + 1
15:    return newP
16: end function
```

2.3. Scattering function. If the new perturbed particle **P** is worse than the current particle **currentP**, the SCATTERING function (see Algorithm 5) is activated. This function uses a Metropolis scheme: with a defined probability, the current particle **currentP** is replaced by a new random solution **P**, or a series of small perturbations are performed on it.

Algorithm 5 Scattering Function

```
1: function SCATTERING(currentP, newP, bestP)
2:     p_scattering = 1−(f(bestP)/f(newP))
3:     if p_scattering > rand(0,1) then
4:         P = RANDOMSOLUTION
5:         NFE = NFE + 1
6:         P = OPPOSITION(P)
7:     else
8:         P = EXPLORATION(currentP)
9:     end if
10:    return P
11: end function
```

2.4. Blackboard updating. Particles in the whole population behave cooperatively, i.e., the best particle overall is over-copied for all other particles in the set, through a blackboard strategy. The UPDATEBLACKBOARD procedure is applied each a number of function evaluations ($N_{FE_{\text{blackboard}}}$).

The master processor receives all the best particles from the others processors, using MPI_SEND for the slave processors, and MPI_RECEIVE fo the master processor (Algorithm 6 – lines 4 and 9). Then, the best overall is

selected, and it is sent to all the processor by mean of the MPI_BROADCAST mechanism (Algorithm 6 – lines 7 and 10).

Algorithm 6 UpdateBlackboard Function

```
1:  function UPDATEBLACKBOARD
2:      if processor == 0 then                          ▷ Master processor selects the best overall particle
3:          for i ← 1 to N_processors do
4:              P = MPI_RECEIVE(i)                      ▷ Receive best particle from each processor
5:              if f(P) < f(bestOverallP) or i == 1 then
6:                  bestOverallP = P                    ▷ Get the best particle overall
7:              end if
8:          end for
9:          MPI_BROADCAST(bestOverallP)                 ▷ Send best particle overall to the other processors
10:     else                                            ▷ Other processors
11:         MPI_SEND(bestP_i)                           ▷ Send the self best particle to the master processor
12:         MPI_BROADCAST(bestOverallP)                 ▷ Receive best particle overall
13:         bestP_i = bestOverallP                      ▷ Substitute the self best particle by the best particle overall
14:     end if
15:     return bestOverallP
16: end function
```

3. Opposition-Based Learning. The OBL concept appeared in 2005 [65, 66]. Thenceforth other variants have been proposed, improving the exploration/exploitation of the search space and improving the convergence.

OBL has been applied to improve the performance of various computational intelligence methods, such as artificial neural networks [69, 32], fuzzy logic [67], stochastic algorithms [1, 19, 26, 27, 28, 44, 49, 53, 71] and miscellaneous applications, introducing the area of Opposition-Based Computing [2, 72].

3.1. Basic concepts. The opposite of a candidate solution is created by reflecting the candidate through the center of the domain [2, 65, 72]. Mathematically, the opposite number x_o of a real number x is defined as below

$$x_o = LB + UB - x \,. \tag{3.1}$$

The opposite point $\mathbf{P}_o(x_{o_1}, x_{o_2}, \cdots, x_{o_D})$ of a point $\mathbf{P}(x_1, x_2, \cdots, x_D)$, with D dimensions, is completely defined by its coordinates,

$$x_{o_d} = \mathbf{LB}_{(d)} + \mathbf{UB}_{(d)} - x_{_d} \tag{3.2}$$

where $x_{_d} \in \mathbb{R}$, with $\mathbf{LB}_{(d)} \leq x_{_d} \leq \mathbf{UB}_{(d)}\ \forall d \in \{1, 2, \cdots, D\}$. $\mathbf{LB}_{(d)}$ and $\mathbf{UB}_{(d)}$ are the lower and upper boundaries for the variable $x_{_d}$, respectively.

Other variants of the OBL have been developed [19, 54, 68]. *Quasi-opposition* reflects a point to a random point between the center of the domain and the opposite point. *Quasi-reflection* projects the point to a random point between the center of the domain and itself. Finally, *Center-based* sampling reflects the point between itself and its opposite. Mathematically, a quasi-opposite point $\mathbf{P}_{qo}$, a quasi-reflective point $\mathbf{P}_{qr}$, and a center-based sampling point $\mathbf{P}_{cb}$, are defined by

$$\mathbf{P}_{qo}(x_{qo_1}, x_{qo_2}, \cdots, x_{qo_D}) \mid x_{qod} = \text{rand}(\mathbf{M}_{(d)}, x_{o_d}) \,, \tag{3.3}$$

$$\mathbf{P}_{qr}(x_{qr_1}, x_{qr_2}, \cdots, x_{qr_D}) \mid x_{qr_d} = \text{rand}(\mathbf{M}_{(d)}, x_{_d}) \,, \tag{3.4}$$

$$\mathbf{P}_{cb}(x_{cb_1}, x_{cb_2}, \cdots, x_{cb_D}) \mid x_{cb_d} = \text{rand}(x_{_d}, x_{o_d}) \,, \tag{3.5}$$

respectively, where $\mathbf{M}_{(d)}$ is the center of the interval $\left[\mathbf{LB}_{(d)}, \mathbf{UB}_{(d)}\right]$, and it can be calculated as $\mathbf{M}_{(d)} = \left(\mathbf{LB}_{(d)} + \mathbf{UB}_{(d)}\right)/2$, and $rand(\cdot)$ is a random number uniformly distributed between the first and second number in the argument. Figure 3.1 shows these concepts graphically for one dimension.

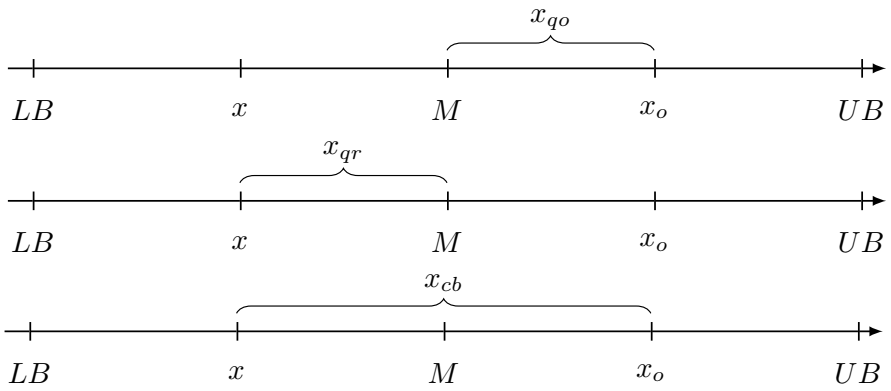


FIGURE 3.1. *Opposition Based Learning for one dimension*

Algorithm 7 shows the computational implementation of these different concepts in OBL. Assuming $f(\cdot)$ is the objective function used to measure the fitness of a candidate solution, the opposition-based optimization consists to apply the procedure: if $f(\mathbf{P}_o) < f(\mathbf{P})$, *i.e.* $\mathbf{P}_o$ has better fitness than $\mathbf{P}$ in a minimization problem, then $\mathbf{P}$ will be replaced by $\mathbf{P}_o$; otherwise, $\mathbf{P}$ will be maintained.

Algorithm 7 Opposition Function

function OPPOSITION(P)
  for $d \leftarrow 1$ **to** D **do**
    Obtain $\mathbf{min}_{(d)}$, $\mathbf{max}_{(d)}$
    $\mathbf{M}_{(d)} = \left(\mathbf{min}_{(d)} + \mathbf{max}_{(d)}\right)/2$
    $\mathbf{OP}_{(d)} = \mathbf{min}_{(d)} + \mathbf{max}_{(d)} - \mathbf{P}_{(d)}$
    r = random number $\in [0, 1]$
    switch type **do**
      case opposition
        $\mathbf{OP}_{(d)} = \mathbf{OP}_{(d)}$
      end case
      case quasi-opposition
        if $\mathbf{P}_{(d)} < \mathbf{M}_{(d)}$ **then**
          $\mathbf{OP}_{(d)} = \mathbf{M}_{(d)} + r\left(\mathbf{OP}_{(d)} - \mathbf{M}_{(d)}\right)$
        else
          $\mathbf{OP}_{(d)} = \mathbf{OP}_{(d)} + r\left(\mathbf{M}_{(d)} - \mathbf{OP}_{(d)}\right)$
        end if
      end case
      case quasi-reflected
        if $\mathbf{P}_{(d)} < \mathbf{M}_{(d)}$ **then**
          $\mathbf{OP}_{(d)} = \mathbf{P}_{(d)} + r\left(\mathbf{M}_{(d)} - \mathbf{P}_{(d)}\right)$
        else
          $\mathbf{OP}_{(d)} = \mathbf{M}_{(d)} + r\left(\mathbf{P}_{(d)} - \mathbf{M}_{(d)}\right)$
        end if
      end case
      case center-based sampling
        if $\mathbf{P}_{(d)} < \mathbf{M}_{(d)}$ **then**
          $\mathbf{OP}_{(d)} = \mathbf{P}_{(d)} + r\left(\mathbf{OP}_{(d)} - \mathbf{P}_{(d)}\right)$
        else
          $\mathbf{OP}_{(d)} = \mathbf{OP}_{(d)} + r\left(\mathbf{P}_{(d)} - \mathbf{OP}_{(d)}\right)$
        end if
      end case
    end switch
  end for
  $NFE = NFE + 1$
  if $f(\mathbf{OP}) < f(\mathbf{P})$ **then**
    $\mathbf{P} = \mathbf{OP}$
  end if
  return P
end function

4. Multi-Particle Collision Algorithm with Opposition-Based Learning. Four new variants of the MPCA are introduced: Oppositional MPCA (OMPCA), Quasi-Oppositional MPCA (QOMPCA), Quasi-Reflective MPCA (QRMPCA), and the Center-Based Sampling MPCA (CBMPCA). The pseudo-code of the computational implementation of these variants are shown in Algorithm 8.

These new versions of the algorithm make possible to perform a more intense exploration of the search space, using the oppositions concepts working together with randomness.

4.1. Opposite Based Population Initialization. Random number generation is commonly the most used choice to create an initial population. By utilizing opposition working together with randomness, it may obtain better-starting candidates even when there is no *a priori* knowledge about the solution.

In the proposed variants of MPCA, the first step is to create the initial solution for each particle as usual. Next, the opposite solution is calculated within the search space $\left[\mathbf{LB}_{(d)}, \mathbf{UB}_{(d)}\right]$. The original solution is substituted by the opposite solution if the latter has a better fitness (see Algorithm 8, line 6).

4.2. Opposite Based Traveling in the search space. The application of OBL based on the traveling of the particles in the search space is dependent on the MPCA function being called.

When a *Perturbation* is applied on a particle, the (new) opposite particle is calculated at the same time.. The best particle among them will be maintained as the "new particle" (see Algorithm 2, line 10). The bounds to create the opposite particle is dynamically reduced to $\left[\mathbf{min}_{(d)}, \mathbf{max}_{(d)}\right]$, where $\mathbf{min}_{(d)}$ and $\mathbf{max}_{(d)}$ are the minimum and maximum values for each dimension in all the population of particles.

When an *Exploration* is performed, the new opposing particle is also calculated, with a jumping rate J_r (see Algorithm 3, lines 5 and 6). The bounds to create the opposite particle is dynamically reduced to $\left[\mathbf{min}_{(d)}, \mathbf{max}_{(d)}\right]$, as was done in the *Perturbation* function.

In the *Scattering* function, if a random particle must be created, then the opposite particle is also created using the original bounds $\left[\mathbf{LB}_{(d)}, \mathbf{UB}_{(d)}\right]$ (see Algorithm 5, line 6).

After applying *Perturbation, Exploration*, and *Scattering* functions to generate the new solution of a particle, then the opposite of the best particle heretofore is calculated using the computed limits $\left[\mathbf{min}_{(d)}, \mathbf{max}_{(d)}\right]$ (see Algorithm 8, line 20).

Algorithm 8 Multi-Particle Collision Algorithm with Opposition-Based Learning

1: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do** ▷ Initial set of particles
2: $N_{FE_i} = 0$, $N_{FE_{\text{lastUpdate}}} = 0$
3: **for** $j \leftarrow 1$ **to** $N_{\text{particles}}$ **do**
4: **currentP**$_{i,j}$ = RANDOMSOLUTION
5: $N_{FE_i} = N_{FE_i} + 1$
6: **currentP**$_{i,j}$ = OPPOSITION(**currentP**$_{i,j}$)
7: **end for**
8: **end for**
9: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do** ▷ Initial blackboard
10: **bestP**$_i$ = UPDATEBLACKBOARD
11: **end for**
12: **while** $N_{FE_i} < N_{FE_{\text{mpca}}}$ **do** ▷ Stopping criteria
13: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do**
14: **for** $j \leftarrow 1$ **to** $N_{\text{particles}}$ **do**
15: **newP**$_{i,j}$ = PERTURBATION(**currentP**$_{i,j}$)
16: **if** f(**newP**$_{i,j}$) $< f$(**currentP**$_{i,j}$) **then**
17: **currentP**$_{i,j}$ = **newP**$_{i,j}$
18: **currentP**$_{i,j}$ = EXPLORATION(**currentP**$_{i,j}$)
19: **else**
20: **currentP**$_{i,j}$ = SCATTERING(**currentP**$_{i,j}$, **newP**$_{i,j}$, **bestP**$_i$)
21: **end if**
22: **if** f(**currentP**$_{i,j}$) $< f$(**bestP**$_i$) **then**
23: **bestP**$_i$ = **currentP**$_{i,j}$
24: **end if**
25: **bestP**$_i$ = OPPOSITION(**bestP**$_i$)
26: **end for**
27: **end for**
28: **if** N_{FE_i} - $N_{FE_{\text{lastUpdate}}} > N_{\text{blackboard}}$ **then**
29: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do** ▷ Final blackboard
30: **bestP**$_i$ = UPDATEBLACKBOARD
31: $N_{FE_{\text{lastUpdate}}} = N_{FE_i}$
32: **end for**
33: **end if**
34: **end while**
35: **for** $i \leftarrow 1$ **to** $N_{\text{processors}}$ **do** ▷ Final blackboard
36: **bestP**$_i$ = UPDATEBLACKBOARD
37: **end for**
38: **return bestP**$_1$

5. Experimental Analysis.

5.1. Benchmark functions. Thirty benchmark functions with different characteristics were implemented to evaluate the performance of MPCA and their variants. Those functions are commonly used in the literature as benchmark functions to evaluate optimization algorithms [3, 19, 30, 41, 46, 47, 50, 70], and they are selected based on different properties of separability and modality, representing a varied range of difficulty. In this selection, we have 10 unimodal, 20 multimodal, 13 separable, and 17 non-separable, as shown in Table 5.1.

TABLE 5.1
Classification of the benchmark functions

	Separable	**Non-separable**
Unimodal	Sphere (f_1) Powell sum (f_2) Sum squares (f_3) Quartic with noise (f_4) Schwefel 2.21 (f_5) Step (f_6)	Schwefel 1.2 (f_7) Dixon & Price (f_8) Schwefel 2.22 (f_9) Rosenbrock (f_{10})
Multi-modal	Schwefel 2.26 (f_{11}) Michalewicz (f_{12}) Rastrigin (f_{13}) Alpine (f_{14}) Levy (f_{15})	Exponential (f_{16}) Rana (f_{17}) Griewank (f_{18}) Ackley (f_{19}) Zakharov (f_{20}) Salomon (f_{21}) Egg holder (f_{22}) Penalized 1 (f_{23}) Penalized 2 (f_{24})

Most of the functions have their global minimum of zero by definition, as the Sphere function, or intentionally normalized, as the Schwefel 2.26 function. Only the Egg holder function in this set have its optima varying with their dimension.

Other two groups of functions is tested: Rotated and shifted version for the Rastrigin, Griewank, and Ackley functions were tested.

Information about all these functions is shown in Table A.1.

5.2. Configuration of the experiments. The fine-tuned control parameters for the MPCA and the variants are shown in Table 5.2. The parameters showed in the Table are different from those used in the literature [43], looking for a better performance for the benchmarks presented.

TABLE 5.2
Control parameters for MPCA

Parameter	Value
$N_{\text{processors}}$	4
$N_{\text{particles}}$	100
$NFE_{\text{blackboard}}$	10000
$NFE_{\text{internalMpca}}$	500
IL	0.7
SL	1.2

All the algorithms are terminated when the number of function evaluation exceeds 3000000 ($NFE_{\text{mpca}} = D \times 10^5$). A trial is considered successfully if the error between the best function value obtained and the optima $\Delta f = 10^{-8}$.

Experiments were performed in a personal computer with 8x Intel® Core™ i7-4790 CPU @ 3.60GHz, with 8 GB of memory, operating with Ubuntu 14.04.3 LTS.

5.3. Results. For each algorithm, a total of 50 trials were performed over each function of the benchmark bed, using the serial and the parallel configurations. Table 5.3 shows the best function values, and Table 5.4 shows the average values over the 50 trials. The best results for each benchmark are shaded.

MPCA and OMPCA failed in the minimization task on many functions ($f_{5-7}, f_9, f_{13}, f_{14}$ and f_{20}), while the other variants could solve the optimization problem. In other functions ($f_{1-3}, f_{16}, f_{18}, f_{19}, f_{21}, f_{28-30}$), QOMPCA, QRMPCA and CBMPCA outperform the values obtained by the MPCA and OMPCA.

The addition of the opposition to the MPCA reports worse results than those obtained with the MPCA. Moreover, generating solutions with quasi-opposition, quasi-reflective and center-based sampling mechanisms improves the results the task of optimizing. This improvement increases, even more, when the parallel versions are used.

Table 5.5 shows the average number of function evaluations used by each algorithm for solving the benchmark functions. Inside the parentheses is shown the success rate of the algorithm (0 implies no success, while 1 means that all the trials were successful). Over other functions ($f_4, f_8, f_{10-12}, f_{15}, f_{17}, f_{22-27}$), none of the algorithms succeeded. In those functions where all the algorithms had success ($f_{1-3}, f_{16}, f_{18}, f_{19}, f_{21}, f_{28-30}$), QOMPCA, QRMPCA and CBMPCA stopped using fewer function evaluations than MPCA and OMPCA.

Figure 5.1 shows the convergence curves of the objective functions f_1, f_7, f_{13} and f_{21} for five random experiments using each algorithm in the serial configuration. In these graphics, we can see that always the canonical MPCA stagnates before the quasi-versions. The use of opposition/reflection accelerates the convergence. The best convergence results were achieved using the QOMPCA and the QRMPCA in almost all cases.

Finally, Table 5.6 shows the running time spent by each algorithm for solving the problems. The time cost for the serial configuration of the quasi-variants (QOMPCA, QRMPCA, and CBMPCA) is, in some functions, up to two orders of magnitude lower than in the serial MPCA and OMPCA or the parallel versions. This evidence that those variants converge to the value to reach in a few functions evaluations.

For the parallel configuration of the quasi-variants, due to the blackboard updating, many function evaluations must be performed before the algorithm stop, even if the minimum function value were reached. That is why they use more function evaluations (and take more running time).

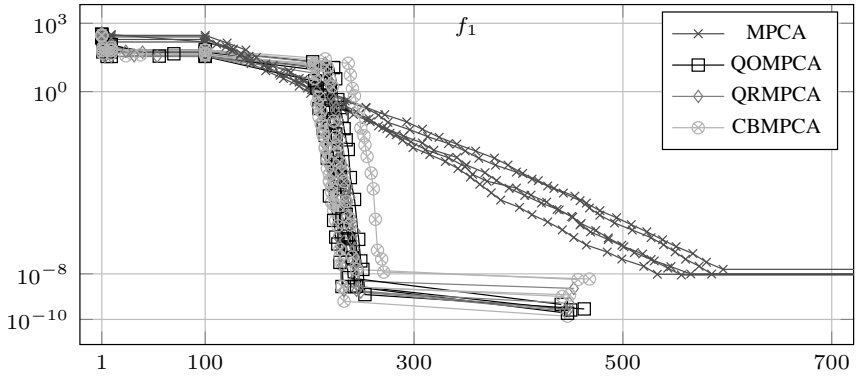
In those functions which the algorithms were unsuccessful, the time spent in the parallel algorithms is a few seconds less than the time consumed by the serial algorithms.

5.3.1. Statistical Analysis. For the multiple non-parametric comparisons among the algorithms we use the Friedman method implemented on the *agricolae*[1] package on R. Besides R itself provides a function `friedman.test()`, it reports only a single value, while *agricolae* performs multiple comparisons [13].

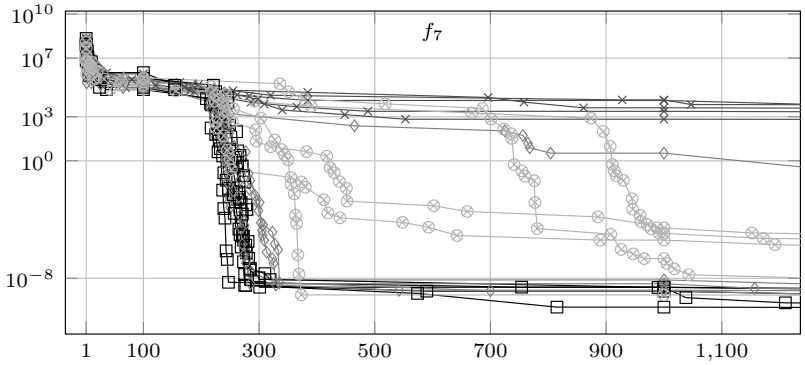
In both configurations (serial and parallel), the Friedman test revealed a significant difference in the average error of the algorithms ($\chi^2 = 14.7$ (serial), $\chi^2 = 13.3$ (parallel), $p < 0.01$). Besides, the test showed significant differences between group A (OMPCA), and B (CBMPCA, QRMPCA, and QOMPCA), having the MPCA in both groups.

A significant difference was found in the running time of the algorithms (χ^2 = 57.5 (serial), χ^2 = 56.5 (parallel), $p < 0.01$). In the test for the serial configuration, QRMPCA belongs to the group C, being the fastest algorithm in convergence. For the parallel configuration, QRMPCA and CBMPCA are included in the group C, while the QOMPCA belongs to both C and B group. In both cases, MPCA was classified as the slowest algorithm.

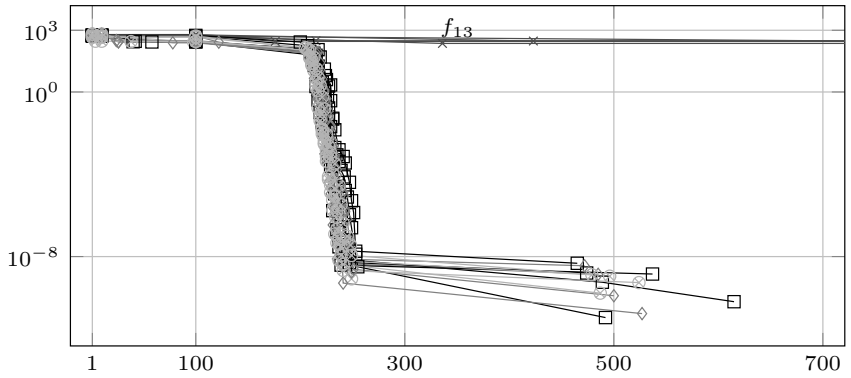
[1] See https://cran.r-project.org/web/packages/agricolae/



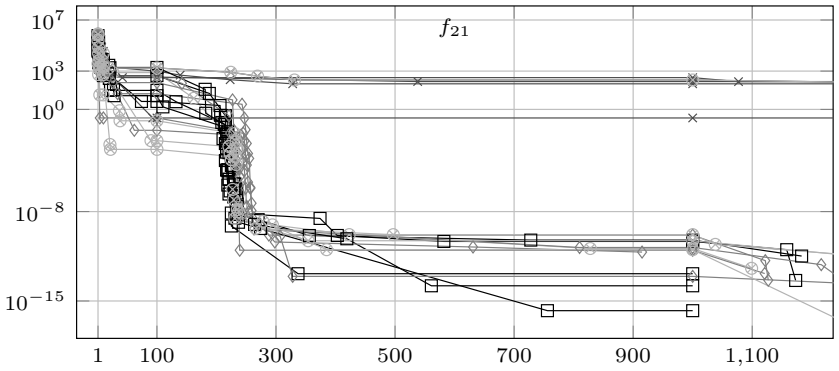
(a) An example of an individual figure sub-caption.



(b) A slightly shorter sub-caption.



(c) A slightly shorter sub-caption.



(d) A slightly shorter sub-caption.

FIGURE 5.1. *Curves of the objective function value vs. number of function evaluation (NFE) for each algorithm in five random experiments*

TABLE 5.3
Best error obtained in the benchmark function bed

	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Serial configuration	f_1	7.73×10^{-9}	7.42×10^{-9}	6.86×10^{-11}	6.18×10^{-11}	1.05×10^{-10}
	f_2	5.06×10^{-9}	5.40×10^{-9}	2.54×10^{-12}	1.77×10^{-12}	1.11×10^{-11}
	f_3	8.36×10^{-9}	8.57×10^{-9}	2.48×10^{-11}	4.00×10^{-11}	9.64×10^{-11}
	f_4	1.12×10^{-2}	1.40×10^{-2}	1.15×10^{-7}	1.71×10^{-7}	5.31×10^{-8}
	f_5	6.92×10^{-4}	1.42×10^{-3}	4.52×10^{-10}	1.21×10^{-10}	1.99×10^{-10}
	f_6	0	0	0	0	0
	f_7	1.50×10^{-2}	1.89×10^{-2}	1.62×10^{-11}	5.78×10^{-13}	3.36×10^{-12}
	f_8	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}
	f_9	6.63×10^{-4}	9.75×10^{-4}	6.71×10^{-10}	1.14×10^{-9}	5.60×10^{-10}
	f_{10}	2.72×10^{1}	2.70×10^{1}	2.81×10^{1}	2.81×10^{1}	2.81×10^{1}
	f_{11}	5.79×10^{3}	5.74×10^{3}	5.72×10^{3}	5.76×10^{3}	5.71×10^{3}
	f_{12}	2.59×10^{-5}	4.3135×10^{-5}	2.03×10^{-4}	3.14×10^{-4}	1.62×10^{-4}
	f_{13}	7.45×10^{1}	7.90×10^{1}	1.66×10^{-10}	1.44×10^{-11}	3.81×10^{-12}
	f_{14}	3.06×10^{0}	4.65×10^{0}	1.75×10^{-9}	1.27×10^{-9}	1.33×10^{-9}
	f_{15}	1.64×10^{-2}	2.34×10^{-2}	2.46×10^{-2}	2.20×10^{-2}	2.43×10^{-2}
	f_{16}	7.74×10^{-9}	7.67×10^{-9}	6.66×10^{-11}	2.01×10^{-10}	1.04×10^{-10}
	f_{17}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}
	f_{18}	7.85×10^{-9}	8.30×10^{-9}	1.23×10^{-10}	2.39×10^{-11}	2.61×10^{-10}
	f_{19}	9.71×10^{-9}	9.99×10^{-9}	7.06×10^{-10}	7.77×10^{-10}	5.41×10^{-10}
	f_{20}	1.20×10^{-8}	4.85×10^{-5}	9.48×10^{-9}	3.73×10^{-9}	5.04×10^{-9}
	f_{21}	6.21×10^{-9}	6.31×10^{-9}	4.74×10^{-15}	1.56×10^{-14}	9.25×10^{-15}
	f_{22}	-4.61×10^{2}	-4.52×10^{2}	-1.16×10^{2}	-1.81×10^{2}	-5.26×10^{2}
	f_{23}	1.67×10^{-1}	5.79×10^{-1}	7.32×10^{-1}	6.10×10^{-1}	6.48×10^{-1}
	f_{24}	1.39×10^{0}	1.68×10^{0}	2.11×10^{0}	2.34×10^{0}	2.32×10^{0}
	f_{25}	2.20×10^{2}	2.46×10^{2}	2.24×10^{2}	1.98×10^{2}	1.83×10^{2}
	f_{26}	7.16×10^{2}	8.04×10^{2}	7.27×10^{2}	9.22×10^{2}	8.07×10^{2}
	f_{27}	1.92×10^{1}	1.71×10^{1}	1.86×10^{1}	1.97×10^{1}	1.95×10^{1}
	f_{28}	5.31×10^{-9}	5.41×10^{-9}	0	0	0
	f_{29}	5.23×10^{-9}	5.31×10^{-9}	7.77×10^{-15}	2.55×10^{-15}	4.44×10^{-16}
	f_{30}	7.21×10^{-9}	7.19×10^{-9}	2.86×10^{-12}	1.10×10^{-11}	1.93×10^{-11}
	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Parallel configuration	f_1	7.27×10^{-9}	7.71×10^{-9}	6.79×10^{-22}	1.68×10^{-21}	5.03×10^{-23}
	f_2	2.78×10^{-9}	3.96×10^{-9}	8.66×10^{-35}	6.75×10^{-36}	4.77×10^{-32}
	f_3	8.46×10^{-9}	8.01×10^{-9}	3.63×10^{-22}	2.99×10^{-21}	2.45×10^{-22}
	f_4	1.14×10^{-2}	1.14×10^{-2}	9.67×10^{-8}	1.11×10^{-7}	3.83×10^{-7}
	f_5	2.07×10^{-3}	1.60×10^{-3}	7.80×10^{-15}	6.52×10^{-15}	9.28×10^{-16}
	f_6	0	0	0	0	0
	f_7	1.51×10^{-3}	2.56×10^{-2}	7.26×10^{-15}	5.33×10^{-15}	4.60×10^{-14}
	f_8	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}
	f_9	6.15×10^{-4}	8.99×10^{-4}	3.27×10^{-13}	8.79×10^{-13}	8.45×10^{-13}
	f_{10}	2.72×10^{1}	2.73×10^{1}	2.73×10^{1}	2.81×10^{1}	2.79×10^{1}
	f_{11}	5.64×10^{3}	5.22×10^{3}	5.92×10^{3}	5.67×10^{3}	5.81×10^{3}
	f_{12}	1.12×10^{-4}	2.83×10^{-4}	4.89×10^{-4}	1.78×10^{-4}	4.03×10^{-4}
	f_{13}	7.63×10^{1}	6.78×10^{1}	0	0	0
	f_{14}	3.86×10^{-1}	5.09×10^{0}	9.71×10^{-14}	1.11×10^{-13}	1.13×10^{-13}
	f_{15}	1.49×10^{-2}	2.31×10^{-2}	2.21×10^{-2}	2.28×10^{-2}	2.43×10^{-2}
	f_{16}	6.48×10^{-9}	7.44×10^{-9}	0	0	0
	f_{17}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}
	f_{18}	8.15×10^{-9}	8.53×10^{-9}	0	0	0
	f_{19}	9.71×10^{-9}	3.68×10^{-5}	4.00×10^{-15}	7.55×10^{-15}	2.18×10^{-14}
	f_{20}	1.48×10^{-8}	1.66×10^{-5}	7.41×10^{-9}	2.99×10^{-9}	3.90×10^{-9}
	f_{21}	5.41×10^{-9}	5.70×10^{-9}	1.96×10^{-29}	1.59×10^{-31}	2.16×10^{-29}
	f_{22}	-2.15×10^{2}	-5.43×10^{2}	-2.35×10^{2}	-3.04×10^{2}	-2.84×10^{2}
	f_{23}	1.97×10^{-1}	6.29×10^{-1}	6.91×10^{-1}	6.77×10^{-1}	6.16×10^{-1}
	f_{24}	1.12×10^{0}	1.42×10^{0}	2.22×10^{0}	2.37×10^{0}	2.13×10^{0}
	f_{25}	2.08×10^{2}	2.10×10^{2}	2.10×10^{2}	2.39×10^{2}	2.24×10^{2}
	f_{26}	4.27×10^{2}	5.56×10^{2}	3.48×10^{2}	5.23×10^{2}	4.62×10^{2}
	f_{27}	1.82×10^{1}	1.71×10^{1}	1.87×10^{1}	1.89×10^{1}	1.79×10^{1}
	f_{28}	4.38×10^{-9}	5.17×10^{-9}	0	0	0
	f_{29}	8.12×10^{-10}	5.19×10^{-9}	0	0	0
	f_{30}	6.17×10^{-9}	7.18×10^{-9}	4.44×10^{-16}	4.44×10^{-16}	4.44×10^{-16}

TABLE 5.4
Average error obtained in the benchmark function bed

	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Serial configuration	f_1	8.41×10^{-9}	8.29×10^{-9}	5.15×10^{-9}	4.29×10^{-9}	4.24×10^{-9}
	f_2	6.30×10^{-9}	6.44×10^{-9}	1.71×10^{-9}	2.35×10^{-9}	2.67×10^{-9}
	f_3	9.44×10^{-9}	9.40×10^{-9}	4.14×10^{-9}	3.86×10^{-9}	4.74×10^{-9}
	f_4	1.77×10^{-2}	2.49×10^{-2}	2.48×10^{-6}	2.46×10^{-6}	2.91×10^{-6}
	f_5	8.44×10^{-3}	1.21×10^{-2}	3.67×10^{-9}	3.65×10^{-9}	3.31×10^{-9}
	f_6	9.40×10^{-1}	1.30×10^{0}	0	0	0
	f_7	2.29×10^{0}	3.76×10^{0}	2.24×10^{-10}	1.48×10^{-10}	4.35×10^{-10}
	f_8	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}
	f_9	1.21×10^{-3}	1.31×10^{-3}	5.96×10^{-9}	5.67×10^{-9}	5.44×10^{-9}
	f_{10}	2.75×10^{1}	2.75×10^{1}	2.81×10^{1}	2.83×10^{1}	2.82×10^{1}
	f_{11}	6.22×10^{3}	6.30×10^{3}	6.34×10^{3}	6.31×10^{3}	6.32×10^{3}
	f_{12}	1.44×10^{-3}	2.33×10^{-3}	2.55×10^{-3}	3.12×10^{-3}	2.62×10^{-3}
	f_{13}	9.72×10^{1}	1.03×10^{2}	3.81×10^{-9}	2.87×10^{-9}	3.25×10^{-9}
	f_{14}	8.24×10^{0}	1.29×10^{1}	5.51×10^{-9}	5.98×10^{-9}	5.46×10^{-9}
	f_{15}	2.29×10^{-2}	3.23×10^{-2}	3.28×10^{-2}	3.21×10^{-2}	3.31×10^{-2}
	f_{16}	8.42×10^{-9}	8.45×10^{-9}	4.05×10^{-9}	5.05×10^{-9}	3.79×10^{-9}
	f_{17}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}
	f_{18}	9.07×10^{-9}	9.28×10^{-9}	4.35×10^{-9}	3.96×10^{-9}	4.46×10^{-9}
	f_{19}	5.63×10^{-5}	6.29×10^{-5}	5.45×10^{-9}	5.61×10^{-9}	5.89×10^{-9}
	f_{20}	3.33×10^{-6}	1.01×10^{-3}	7.75×10^{-7}	1.04×10^{-8}	1.42×10^{-8}
	f_{21}	8.62×10^{-9}	8.67×10^{-9}	3.10×10^{-11}	3.42×10^{-11}	1.96×10^{-11}
	f_{22}	5.20×10^{2}	6.45×10^{2}	7.49×10^{2}	6.90×10^{2}	7.05×10^{2}
	f_{23}	3.14×10^{-1}	7.06×10^{-1}	8.69×10^{-1}	7.94×10^{-1}	7.98×10^{-1}
	f_{24}	2.10×10^{0}	2.09×10^{0}	2.48×10^{0}	2.61×10^{0}	2.52×10^{0}
	f_{25}	2.63×10^{2}	2.76×10^{2}	2.57×10^{2}	2.64×10^{2}	2.16×10^{2}
	f_{26}	7.52×10^{2}	8.28×10^{2}	7.57×10^{2}	9.62×10^{2}	8.29×10^{2}
	f_{27}	2.00×10^{1}	1.89×10^{1}	1.99×10^{1}	2.01×10^{1}	2.02×10^{1}
	f_{28}	6.69×10^{-9}	6.83×10^{-9}	4.47×10^{-11}	3.52×10^{-11}	1.62×10^{-11}
	f_{29}	6.30×10^{-9}	6.28×10^{-9}	7.47×10^{-11}	8.56×10^{-11}	1.25×10^{-10}
	f_{30}	7.76×10^{-9}	7.69×10^{-9}	3.43×10^{-10}	3.58×10^{-10}	3.93×10^{-10}
	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Parallel configuration	f_1	7.91×10^{-9}	8.34×10^{-9}	4.94×10^{-18}	7.73×10^{-18}	3.63×10^{-18}
	f_2	4.17×10^{-9}	5.42×10^{-9}	3.02×10^{-26}	2.90×10^{-26}	5.67×10^{-26}
	f_3	9.49×10^{-9}	9.42×10^{-9}	1.05×10^{-17}	1.61×10^{-17}	1.42×10^{-17}
	f_4	1.79×10^{-2}	2.52×10^{-2}	2.51×10^{-6}	2.45×10^{-6}	3.10×10^{-6}
	f_5	7.57×10^{-3}	1.55×10^{-2}	7.23×10^{-13}	5.79×10^{-13}	7.46×10^{-13}
	f_6	1.02×10^{0}	1.34×10^{0}	0	0	0
	f_7	1.71×10^{0}	6.65×10^{0}	1.84×10^{-12}	1.37×10^{-12}	7.19×10^{-11}
	f_8	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}	6.67×10^{-1}
	f_9	1.16×10^{-3}	1.29×10^{-3}	1.24×10^{-11}	1.95×10^{-11}	1.50×10^{-11}
	f_{10}	2.75×10^{1}	2.75×10^{1}	2.81×10^{1}	2.83×10^{1}	2.82×10^{1}
	f_{11}	6.26×10^{3}	6.22×10^{3}	6.37×10^{3}	6.33×10^{3}	6.37×10^{3}
	f_{12}	1.45×10^{-3}	2.22×10^{-3}	3.31×10^{-3}	3.57×10^{-3}	2.50×10^{-3}
	f_{13}	9.94×10^{1}	1.06×10^{2}	0	0	0
	f_{14}	7.77×10^{0}	1.30×10^{1}	3.66×10^{-12}	2.12×10^{-12}	2.29×10^{-12}
	f_{15}	2.24×10^{-2}	3.17×10^{-2}	3.23×10^{-2}	3.25×10^{-2}	3.30×10^{-2}
	f_{16}	7.59×10^{-9}	8.30×10^{-9}	0	0	0
	f_{17}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}	4.99×10^{2}
	f_{18}	8.96×10^{-9}	9.16×10^{-9}	6.44×10^{-17}	1.78×10^{-17}	4.66×10^{-17}
	f_{19}	5.72×10^{-5}	6.13×10^{-5}	1.09×10^{-12}	1.07×10^{-12}	7.39×10^{-13}
	f_{20}	2.70×10^{-6}	1.09×10^{-3}	6.79×10^{-7}	1.13×10^{-8}	1.30×10^{-8}
	f_{21}	8.12×10^{-9}	8.57×10^{-9}	1.04×10^{-23}	1.95×10^{-23}	1.35×10^{-23}
	f_{22}	5.61×10^{2}	6.00×10^{2}	7.37×10^{2}	7.31×10^{2}	7.76×10^{2}
	f_{23}	3.25×10^{-1}	7.16×10^{-1}	8.70×10^{-1}	8.04×10^{-1}	7.88×10^{-1}
	f_{24}	2.08×10^{0}	2.04×10^{0}	2.47×10^{0}	2.61×10^{0}	2.52×10^{0}
	f_{25}	2.43×10^{2}	2.49×10^{2}	2.49×10^{2}	2.78×10^{2}	2.53×10^{2}
	f_{26}	4.57×10^{2}	6.08×10^{2}	3.73×10^{2}	5.560×10^{2}	4.87×10^{2}
	f_{27}	1.95×10^{1}	1.85×10^{1}	1.98×10^{1}	1.98×10^{1}	1.97×10^{1}
	f_{28}	5.98×10^{-9}	6.32×10^{-9}	0	0	0
	f_{29}	5.27×10^{-9}	6.06×10^{-9}	7.11×10^{-11}	0	0
	f_{30}	6.93×10^{-9}	7.66×10^{-9}	9.41×10^{-16}	7.99×10^{-16}	1.87×10^{-15}

TABLE 5.5
Average number of function evaluations and success rate (in parenthesis) obtained in the benchmark function bed

	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Serial configuration	f_1	24065 (1)	47314 (1)	546 (1)	546 (1)	546 (1)
	f_2	10226 (1)	19852 (1)	677 (1)	727 (1)	644 (1)
	f_3	61537 (1)	116771 (1)	555 (1)	551 (1)	552 (1)
	f_4	- (0)	- (0)	- (0)	- (0)	- (0)
	f_5	- (0)	- (0)	786 (1)	775 (1)	811 (1)
	f_6	** (.18)	- (.06)	533 (1)	530 (1)	531 (1)
	f_7	- (0)	- (0)	6292 (1)	6355 (1)	14502 (1)
	f_8	- (0)	- (0)	- (0)	- (0)	- (0)
	f_9	- (0)	- (0)	4576 (1)	4295 (1)	4476 (1)
	f_{10}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{11}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{12}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{13}	- (0)	- (0)	612 (1)	608 (1)	610 (1)
	f_{14}	- (0)	- (0)	613 (1)	605 (1)	635 (1)
	f_{15}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{16}	19747 (1)	38705 (1)	577 (1)	578 (1)	580 (1)
	f_{17}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{18}	27697 (1)	57270 (1)	558 (1)	560 (1)	574 (1)
	f_{19}	- (.02)	- (.02)	638 (1)	637 (1)	641 (1)
	f_{20}	- (0)	- (0)	- (.04)	1596562 (.72)	1861391 (.58)
	f_{21}	43568 (1)	84149 (1)	1262 (1)	1234 (1)	1243 (1)
	f_{22}	- (.10)	- (.04)	- (.02)	- (.04)	- (.06)
	f_{23}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{24}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{25}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{26}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{27}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{28}	21960 (1)	42706 (1)	1538 (1)	1477 (1)	1521 (1)
	f_{29}	17244 (1)	34883 (1)	2066 (1)	2156 (1)	2021 (1)
	f_{30}	18999 (1)	37172 (1)	1496 (1)	1480 (1)	1428 (1)
	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Parallel configuration	f_1	50178 (1)	53483 (1)	41194 (1)	41276 (1)	41268 (1)
	f_2	43643 (1)	46980 (1)	41078 (1)	41077 (1)	41093 (1)
	f_3	57941 (1)	135751 (1)	41363 (1)	41331 (1)	41449 (1)
	f_4	- (0)	- (.04)	- (0)	- (0)	- (0)
	f_5	- (0)	- (.04)	41680 (1)	41655 (1)	41799 (1)
	f_6	** (.12)	- (.06)	55418 (1)	55679 (1)	55962 (1)
	f_7	- (0)	- (0)	44155 (1)	44694 (1)	50834 (1)
	f_8	- (0)	- (.06)	- (0)	- (0)	- (0)
	f_9	- (0)	- (0)	41538 (1)	41449 (1)	41586 (1)
	f_{10}	- (0)	- (.02)	- (0)	- (0)	- (0)
	f_{11}	- (0)	- (0)	- (0)	- (.02)	- (0)
	f_{12}	- (0)	- (0)	- (.04)	- (0)	- (0)
	f_{13}	- (0)	- (.14)	44097 (1)	44162 (1)	43731 (1)
	f_{14}	- (.02)	- (.14)	41637 (1)	41517 (1)	41604 (1)
	f_{15}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{16}	45650 (1)	47694 (1)	42375 (1)	42299 (1)	42270 (1)
	f_{17}	- (0)	- (0)	- (.02)	- (0)	- (.02)
	f_{18}	47474 (1)	69806 (1)	41847 (1)	41888 (1)	41834 (1)
	f_{19}	- (.02)	- (0)	41569 (1)	41655 (1)	41616 (1)
	f_{20}	- (0)	- (0)	- (.06)	1753410 (.64)	1952101 (.60)
	f_{21}	63107 (1)	101810 (1)	41327 (1)	41278 (1)	41326 (1)
	f_{22}	** (.12)	- (.10)	- (.10)	- (.04)	- (.02)
	f_{23}	- (0)	- (.04)	- (0)	- (0)	- (0)
	f_{24}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{25}	- (0)	- (.08)	- (.02)	- (0)	- (.10)
	f_{26}	- (.08)	** (.56)	** (.44)	** (.56)	- (0)
	f_{27}	- (0)	- (0)	- (0)	- (0)	- (0)
	f_{28}	49474 (1)	53972 (1)	46044 (1)	45902 (1)	45669 (1)
	f_{29}	46108 (1)	50998 (1)	42805 (1)	43558 (1)	43744 (1)
	f_{30}	46318 (1)	50381 (1)	41899 (1)	41638 (1)	41665 (1)

TABLE 5.6
Average running time (in seconds) obtained on the benchmark function bed

	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Serial configuration	f_1	3.01×10^{-2}	3.72×10^{-2}	4.56×10^{-4}	4.28×10^{-4}	4.76×10^{-4}
	f_2	1.69×10^{-2}	1.82×10^{-2}	5.82×10^{-4}	5.84×10^{-4}	5.91×10^{-4}
	f_3	7.85×10^{-2}	9.18×10^{-2}	4.84×10^{-4}	4.73×10^{-4}	5.28×10^{-4}
	f_4	3.71×10^{0}	2.47×10^{0}	2.50×10^{0}	2.39×10^{0}	2.41×10^{0}
	f_5	3.32×10^{0}	2.15×10^{0}	7.44×10^{-4}	6.98×10^{-4}	7.92×10^{-4}
	f_6	3.10×10^{0}	2.11×10^{0}	5.76×10^{-4}	4.28×10^{-4}	4.77×10^{-4}
	f_7	6.10×10^{0}	4.93×10^{0}	1.22×10^{-2}	1.13×10^{-2}	2.58×10^{-2}
	f_8	3.47×10^{0}	2.30×10^{0}	2.26×10^{0}	2.19×10^{0}	2.19×10^{0}
	f_9	3.40×10^{0}	2.22×10^{0}	3.79×10^{-3}	2.84×10^{-3}	2.88×10^{-3}
	f_{10}	3.71×10^{0}	2.42×10^{0}	2.19×10^{0}	2.06×10^{0}	2.09×10^{0}
	f_{11}	6.99×10^{0}	5.06×10^{0}	6.34×10^{0}	6.31×10^{0}	5.89×10^{0}
	f_{12}	9.11×10^{0}	7.39×10^{0}	8.41×10^{0}	8.16×10^{0}	8.25×10^{0}
	f_{13}	6.26×10^{0}	4.43×10^{0}	1.14×10^{-3}	1.14×10^{-3}	1.19×10^{-3}
	f_{14}	5.82×10^{0}	4.13×10^{0}	1.03×10^{-3}	1.05×10^{-3}	1.11×10^{-3}
	f_{15}	7.87×10^{0}	6.13×10^{0}	7.19×10^{0}	6.52×10^{0}	6.53×10^{0}
	f_{16}	2.58×10^{-2}	3.40×10^{-2}	5.15×10^{-4}	5.36×10^{-4}	5.79×10^{-4}
	f_{17}	1.52×10^{1}	1.17×10^{1}	1.47×10^{1}	1.42×10^{1}	1.43×10^{1}
	f_{18}	5.06×10^{-2}	9.14×10^{-2}	1.16×10^{-3}	1.25×10^{-3}	1.24×10^{-3}
	f_{19}	5.70×10^{0}	4.21×10^{0}	1.23×10^{-3}	1.30×10^{-3}	1.32×10^{-3}
	f_{20}	3.39×10^{0}	2.14×10^{0}	2.51×10^{0}	1.35×10^{0}	1.58×10^{0}
	f_{21}	5.07×10^{-2}	6.32×10^{-2}	1.24×10^{-3}	1.18×10^{-3}	1.26×10^{-3}
	f_{22}	1.01×10^{1}	8.03×10^{0}	9.90×10^{0}	9.42×10^{0}	9.50×10^{0}
	f_{23}	8.30×10^{0}	6.56×10^{0}	7.97×10^{0}	7.76×10^{0}	7.83×10^{0}
	f_{24}	8.83×10^{0}	6.95×10^{0}	7.37×10^{0}	7.12×10^{0}	7.18×10^{0}
	f_{25}	6.46×10^{0}	4.63×10^{0}	5.38×10^{0}	5.34×10^{0}	5.43×10^{0}
	f_{26}	7.26×10^{0}	5.43×10^{0}	6.51×10^{0}	6.24×10^{0}	6.37×10^{0}
	f_{27}	6.58×10^{0}	4.82×10^{0}	5.81×10^{0}	5.57×10^{0}	5.73×10^{0}
	f_{28}	9.15×10^{-2}	1.63×10^{-1}	5.77×10^{-3}	5.62×10^{-3}	5.69×10^{-3}
	f_{29}	7.13×10^{-2}	1.29×10^{-1}	7.96×10^{-3}	8.00×10^{-3}	7.63×10^{-3}
	f_{30}	8.52×10^{-2}	1.45×10^{-1}	5.90×10^{-3}	5.74×10^{-3}	5.53×10^{-3}
	f	MPCA	OMPCA	QOMPCA	QRMPCA	CBMPCA
Parallel configuration	f_1	2.86×10^{-2}	1.89×10^{-2}	1.25×10^{-2}	1.56×10^{-2}	1.02×10^{-2}
	f_2	2.58×10^{-2}	1.95×10^{-2}	1.21×10^{-2}	1.44×10^{-2}	1.52×10^{-2}
	f_3	3.23×10^{-2}	4.87×10^{-2}	1.67×10^{-2}	1.72×10^{-2}	1.76×10^{-2}
	f_4	1.39×10^{0}	1.11×10^{0}	1.10×10^{0}	1.09×10^{0}	9.63×10^{-1}
	f_5	1.64×10^{0}	9.56×10^{-1}	1.74×10^{-2}	1.72×10^{-2}	1.19×10^{-2}
	f_6	1.34×10^{0}	1.05×10^{0}	1.70×10^{-2}	2.42×10^{-2}	1.53×10^{-2}
	f_7	2.45×10^{0}	2.06×10^{0}	3.18×10^{-2}	3.17×10^{-2}	3.80×10^{-2}
	f_8	1.65×10^{0}	1.18×10^{0}	9.72×10^{-1}	9.92×10^{-1}	6.39×10^{-1}
	f_9	1.20×10^{0}	1.04×10^{0}	1.56×10^{-2}	1.55×10^{-2}	1.35×10^{-2}
	f_{10}	1.24×10^{0}	1.15×10^{0}	8.91×10^{-1}	9.54×10^{-1}	6.06×10^{-1}
	f_{11}	2.70×10^{0}	2.02×10^{0}	2.42×10^{0}	2.42×10^{0}	2.13×10^{0}
	f_{12}	3.40×10^{0}	2.64×10^{0}	3.32×10^{0}	3.30×10^{0}	3.29×10^{0}
	f_{13}	2.72×10^{0}	2.04×10^{0}	2.56×10^{-2}	2.53×10^{-2}	2.54×10^{-2}
	f_{14}	2.40×10^{0}	1.98×10^{0}	2.36×10^{-2}	1.89×10^{-2}	2.35×10^{-2}
	f_{15}	2.46×10^{0}	2.30×10^{0}	2.51×10^{0}	2.51×10^{0}	2.21×10^{0}
	f_{16}	2.34×10^{-2}	1.11×10^{-2}	1.70×10^{-2}	1.77×10^{-2}	1.79×10^{-2}
	f_{17}	5.62×10^{0}	5.11×10^{0}	5.52×10^{0}	5.53×10^{0}	4.65×10^{0}
	f_{18}	4.05×10^{-2}	4.76×10^{-2}	2.83×10^{-2}	2.77×10^{-2}	2.90×10^{-2}
	f_{19}	2.25×10^{0}	1.89×10^{0}	2.48×10^{-2}	2.48×10^{-2}	2.23×10^{-2}
	f_{20}	1.69×10^{0}	1.09×10^{0}	1.19×10^{0}	7.56×10^{-1}	7.81×10^{-1}
	f_{21}	3.54×10^{-2}	3.47×10^{-2}	1.71×10^{-2}	1.67×10^{-2}	1.74×10^{-2}
	f_{22}	3.77×10^{0}	3.03×10^{0}	3.64×10^{0}	3.77×10^{0}	3.88×10^{0}
	f_{23}	3.54×10^{0}	2.80×10^{0}	3.00×10^{0}	3.07×10^{0}	3.08×10^{0}
	f_{24}	3.66×10^{0}	2.88×10^{0}	2.83×10^{0}	2.80×10^{0}	2.62×10^{0}
	f_{25}	2.80×10^{0}	2.15×10^{0}	2.43×10^{0}	2.47×10^{0}	2.14×10^{0}
	f_{26}	2.96×10^{0}	2.12×10^{0}	2.59×10^{0}	2.11×10^{0}	2.58×10^{0}
	f_{27}	2.83×10^{0}	1.90×10^{0}	2.29×10^{0}	2.10×10^{0}	1.90×10^{0}
	f_{28}	9.73×10^{-2}	9.97×10^{-2}	8.87×10^{-2}	8.46×10^{-2}	8.56×10^{-2}
	f_{29}	9.44×10^{-2}	8.46×10^{-2}	8.28×10^{-2}	8.00×10^{-2}	8.12×10^{-2}
	f_{30}	9.76×10^{-2}	9.24×10^{-2}	7.75×10^{-2}	7.63×10^{-2}	7.77×10^{-2}

5.4. Comparison with some metaheuristic algorithms. MPCA and QRMPCA are compared with some metaheuristics: Particle Swarm Optimization (PSO) [61]; Differential Evolution (DE) [51]; Artificial Bee Colony (ABC) [34]; Continuous Genetic Algorithm (CGA) [9]; and Gradient Evolution (GE) [38].

The results from these algorithms over twelve objective functions ($f_{1-3}, f_6, f_7, f_{9-11}, f_{13}, f_{18}, f_{19}, f_{21}$) are all reproduced from the paper [38], obtained using the same control parameters and stopping criteria that were used in the original works. In this experiments, the stopping criterium for the MPCA variants is the number of function evaluations, which is set to 3000000 ($N_{FE_{\text{mpca}}} = D \times 10^5$). For each algorithm were performed 30 experiments. Results below 1×10^{-308} are considered as zero.

Table 5.7 shows the results. The symbol (§) in the cells represents the best results obtained in the original paper [38], and the shaded cells represent the best results overall, comparting our result with the obtained in the mentioned paper. MPCA could not obtain the best results in any case. QRMPCA obtained the best results for all the functions, except for f_{10} – Rosenbrock and f_{11} – Schwefel 2.26, where the ABC and the DE obtained the best results, respectively. For the function f_6, QRMPCA obtained the same results than GE, DE, ABC, and CGA. Also, QRMPCA achieves the best results equal to GE for functions f_{13} and f_{18}, while ties with DE for the f_2 and f_{18}.

Further, MPCA and QRMPCA are compared with some DE, PSO, and ABC variants which are taken from the original paper [24]. Those algorithms are: the basic DE [63]; self-adapting control parameters in DE (jDE) [8]; DE with strategy adaptation (SaDE) [52]; adaptive DE with optional external archive (JADE) [75]; basic PSO [36]; "fully informed" PSO (FIPS) [45]; the self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) [55]; the comprehensive learning PSO (CLPSO) [40]; orthogonal learning PSO (OLPSO-G) [74]; rosenbrock ABC (RABC) [33]; modified ABC (MABC) [21]; powell ABC (PABC) [23]; orthogonal learning GABC (OGABC) [22]; and bare bones ABC (BCABC) [24].

Table 5.8, Table 5.9, and Table 5.10 shows the results of the comparison. The results of the compared DEs, PSOs, and ABCs are come from the paper [24]. The shaded cells represent the best results. QRMPCA outperforms all the other algorithms in six functions, while obtained the worst results overall in four functions.

6. Conclusions. Some variants for the MPCA using the concepts of OBL were presented and tested. Those algorithms use quasi-opposition-based learning (QOBL) and quasi-reflection-based learning (QRBL) concepts mixed with randomness. QOMPCA, QRMPCA, CBMPCA have obtained better results than MPCA and OMPCA, over a benchmark bed with thirty functions of different complexity, with dimension 30.

MPCA was initially codified in a parallel version. In this work, a serial version was also codified. Here, it was shown that the sequential configuration could achieve success with a low computational cost. As expected, the parallel configuration obtained the results in less time than the serial.

The MPCA and QRMPCA were compared with five metaheuristics over 12 functions. MPCA obtained the same or better results in 10 functions than the other algorithms.

In further studies, experiments with real-world problems will be performed, such as solving the Vibration-based Damage Identification problem, and performing the automatic configuration of Artificial Neural Networks for data assimilation and climate prediction. Other variants and configurations of the MPCA will also be implemented and tested.

Acknowledgements. The authors want to acknowledge the National Counsel of Technological and Scientific Development (CNPq) from Brazil for the support to this work [grant number 159547/2013-0].

ORCID and License

Reynier Hernández Torres http://orcid.org/0000-0003-1554-5145,
Haroldo Fraga de Campos Velho http://orcid.org/0000-0003-4968-5330,
Eduardo Fávero Pacheco da Luz https://orcid.org/0000-0003-2081-1831.

TABLE 5.7

Results of the GE, PSO, DE, ABC, CGA, MPCA and QRMPCA for problems in 30D. Results taken from [38]. The symbol (§) represents the best results obtained in the original paper from [38]. The shaded cells represent the best results overall.

	GE	PSO	DE	ABC	CGA	MPCA	QRMPCA
f_1 (Sphere)	2.56×10^{-30}	1.68×10^{-29}	6.32×10^{-157}(§)	4.41×10^{-10}	3.63	4.79×10^{-11}	0
	(2.85×10^{-30})	(5.19×10^{-29})	(0)	(5.42×10^{-10})	(4.02×10^{-1})	(1.18×10^{-11})	(0)
f_2 (Powell sum)	8.66×10^{-67}	1.04×10^{-50}	0 (§)	1.77×10^{-11}	1.27×10^{-1}	2.18×10^{-26}	0
	(3.34×10^{-66})	(5.52×10^{-50})	(0)	(2.99×10^{-11})	(7.11×10^{-2})	(1.66×10^{-26})	(0)
f_3 (Sum squares)	9.07×10^{-32}	4.07×10^{-31}	1.65×10^{-158}(§)	1.28×10^{-8}	4.33×10^{1}	2.65×10^{-9}	0
	(6.45×10^{-1})	(2.30×10^{-1})	(3.38×10^{-2})	(3.09×10^{-1})	(8.06)	(7.28×10^{-10})	(0)
f_6 (Step)	0 (§)	3.33×10^{-2}	0 (§)	0 (§)	0 (§)	9.33×10^{-1}	0
	(0)	(1.83×10^{-1})	(0)	(0)	(0)	(0)	(0)
f_7 (Schwefel 1.2)	6.55×10^{-16}(§)	2.87×10^{1}	8.76×10^{2}	2.23×10^{-7}	2.05×10^{4}	1.46	0
	(5.95×10^{-16})	(2.97×10^{1})	(3.53×10^{2})	(3.54×10^{-7})	(7.91×10^{3})	(2.20)	(0)
f_9 (Schwefel 2.22)	2.58×10^{-17}	6.84×10^{-21}	5.96×10^{-92}(§)	7.69×10^{-5}	7.72	1.12×10^{-3}	1.92×10^{-212}
	(1.72×10^{-17})	(1.64×10^{-20})	(5.70×10^{-92})	(9.09×10^{-5})	(7.31×10^{-1})	(1.67×10^{-4})	(0)
f_{10} (Rosenbrock)	1.81×10^{1}	2.44×10^{1}	2.22×10^{1}	9.05×10^{-8}(§)	1.74×10^{2}	2.75×10^{1}	2.83×10^{1}
	(1.66×10^{-1})	(1.04×10^{1})	(1.31)	(1.49×10^{-7})	(1.08×10^{1})	(1.62×10^{-1})	(2.39×10^{-1})
f_{11} (Schwefel 2.26)	5.49×10^{2}	7.14×10^{3}	3.82×10^{-4}(§)	1.24×10^{4}	6.65×10^{3}	6.20×10^{3}	6.41×10^{3}
	(2.03×10^{2})	(7.38×10^{2})	(2.76×10^{-19})	(5.48×10^{2})	(7.22×10^{2})	(2.22×10^{2})	(2.00×10^{2})
f_{13} (Rastrigin)	0 (§)	2.12×10^{1}	1.99×10^{-1}	1.71×10^{-7}	3.17×10^{1}	9.90×10^{1}	0
	(0)	(4.00)	(4.05×10^{-1})	(3.85×10^{-7})	(1.18×10^{1})	(1.15×10^{1})	(0)
f_{18} (Griewank)	0 (§)	1.45×10^{-2}	0 (§)	2.45×10^{-11}	1.81	1.21×10^{-9}	0
	(0)	(1.72×10^{-2})	(0)	(4.74×10^{-11})	(1.03)	(2.16×10^{-10})	(0)
f_{19} (Ackley)	4.44×10^{-15}(§)	2.03×10^{-14}	7.64×10^{-15}	1.29×10^{-5}	2.46	5.72×10^{-5}	4.44×10^{-16}
	(1.60×10^{-30})	(6.71×10^{-15})	(1.08×10^{-15})	(1.25×10^{-5})	(2.98×10^{-1})	(1.03×10^{-5})	(0)
f_{21} (Salomon)	9.99×10^{-2}	3.77×10^{-1}	2.00×10^{-1}	5.11×10^{-6}(§)	3.00×10^{-1}	1.05×10^{-27}	0
	(4.81×10^{-10})	(6.79×10^{-2})	(1.92×10^{-8})	(4.45×10^{-6})	(2.80×10^{-12})	(2.20×10^{-27})	(0)

TABLE 5.8

Comparisons between MPCA variants and DEs on optimizing 30-dimensional functions. Results were taken from [24]. The shaded cells represent the best results overall.

Fun	Max.FEs	DE	jDE	JADE	SaDE	MPCA	QRMPCA
f_1 Sphere	150000	9.8×10^{-14}	1.46×10^{-28}	2.69×10^{-56}	3.28×10^{-20}	4.79×10^{-139}	0
		(8.4×10^{-14})	(1.78×10^{-28})	(1.41×10^{-55})	(3.63×10^{-20})	(2.01×10^{-140})	(0)
f_9 Schwefel 2.22	200000	1.6×10^{-9}	9.02×10^{-24}	3.18×10^{-25}	3.51×10^{-25}	1.41×10^{-68}	0
		(1.1×10^{-9})	(6.01×10^{-24})	(2.05×10^{-24})	(2.74×10^{-25})	(4.74×10^{-69})	(0)
f_{10} Rosenbrock	300000	2.1×10^{0}	1.3×10^{1}	3.2×10^{-1}	2.1×10^{1}	2.81×10^{1}	2.87×10^{1}
		(1.5×10^{0})	(1.4×10^{1})	(1.1×10^{0})	(7.8×10^{0})	(2.15×10^{-1})	(4.64×10^{-2})
f_6 Step	10000	4.7×10^{3}	6.13×10^{2}	5.62×10^{0}	5.07×10^{1}	0	0
		(1.1×10^{3})	(1.72×10^{2})	(1.87×10^{0})	(1.34×10^{1})	(0)	(0)
f_{11} Schwefel 2.26	100000	5.9×10^{3}	1.70×10^{-10}	2.62×10^{-4}	1.13×10^{-8}	7.56×10^{3}	7.88×10^{3}
		(1.1×10^{3})	(1.71×10^{-10})	(3.59×10^{-4})	(1.08×10^{-8})	(2.59×10^{2})	(3.40×10^{2})
f_{13} Rastrigin	100000	1.8×10^{2}	3.32×10^{-4}	1.33×10^{-1}	2.43×10^{0}	0	0
		(1.3×10^{1})	(6.39×10^{-4})	(9.74×10^{-2})	(1.60×10^{0})	(0)	(0)
f_{19} Ackley	50000	1.1×10^{-1}	2.37×10^{-4}	3.35×10^{-9}	3.81×10^{-6}	4.00×10^{-15}	4.44×10^{-16}
		(3.9×10^{-2})	(7.10×10^{-5})	(2.84×10^{-9})	(8.26×10^{-7})	(0)	(0)
f_{18} Griewank	50000	2.0×10^{-1}	7.29×10^{-6}	1.57×10^{-8}	2.52×10^{-9}	0	0
		(1.1×10^{-1})	(1.05×10^{-5})	(1.09×10^{-7})	(1.24×10^{-8})	(0)	(0)
f_{23} Penalized 1	50000	1.2×10^{-2}	7.03×10^{-8}	1.67×10^{-15}	8.25×10^{-12}	7.22×10^{-1}	1.03×10^{0}
		(1.0×10^{-2})	(5.74×10^{-8})	(1.02×10^{-14})	(5.12×10^{-12})	(5.32×10^{-2})	(8.23×10^{-2})
f_{24} Penalized 2	50000	7.5×10^{-2}	1.80×10^{-5}	1.87×10^{-10}	1.93×10^{-9}	2.76×10^{0}	2.87×10^{0}
		(3.8×10^{-2})	(1.42×10^{-5})	(1.09×10^{-9})	(1.53×10^{-9})	(8.97×10^{-2})	(9.74×10^{-2})
f_{14} Alpine	300000	2.3×10^{-4}	6.08×10^{-10}	2.78×10^{-5}	2.94×10^{-6}	4.16×10^{-72}	0
		(1.7×10^{-4})	(8.36×10^{-10})	(8.43×10^{-6})	(3.47×10^{-6})	(2.16×10^{-72})	(0)

TABLE 5.9

Comparisons between MPCA variants and PSOs on optimizing 30-dimensional functions. Results were taken from [24]. The shaded cells represent the best results overall.

Fun	PSO	FIPS	HPSO-TVAC	CLPSO	OLPSO-G	MPCA	QRMPCA
Sphere	3.34×10^{-14}	2.42×10^{-13}	2.83×10^{-33}	1.58×10^{-12}	4.12×10^{-54}	1.02×10^{-145}	0
	(5.39×10^{-14})	(1.73×10^{-13})	(3.19×10^{-33})	(7.70×10^{-13})	(6.34×10^{-54})	(5.11×10^{-145})	(0)
Schwefel 2.22	1.70×10^{-10}	2.76×10^{-8}	9.03×10^{-20}	2.51×10^{-8}	9.85×10^{-30}	1.82×10^{-71}	0
	(1.39×10^{-10})	(9.04×10^{-9})	(9.58×10^{-20})	(5.84×10^{-9})	(1.01×10^{-29})	(6.17×10^{-72})	(0)
Rosenbrock	2.80×10^{1}	2.51×10^{1}	2.39×10^{1}	1.13×10^{1}	2.15×10^{1}	2.78×10^{1}	2.85×10^{1}
	(2.17×10^{1})	(5.10×10^{-1})	(2.65×10^{1})	(9.85×10^{0})	(2.99×10^{1})	(1.17×10^{-1})	(3.12×10^{-1})
Step	0	0	0	0	0	0	0
	(0)	(0)	(0)	(0)	(0)	(0)	(0)
Schwefel 2.26	3.16×10^{3}	9.93×10^{2}	1.59×10^{3}	3.82×10^{-4}	3.84×10^{2}	6.73×10^{3}	6.88×10^{3}
	(4.06×10^{2})	(5.09×10^{2})	(3.26×10^{2})	(1.28×10^{-5})	(2.17×10^{2})	(2.45×10^{2})	(3.30×10^{2})
Rastrigin	3.57×10^{1}	6.51×10^{1}	9.43×10^{0}	9.09×10^{-5}	1.07×10^{0}	0	0
	(6.89×10^{0})	(1.33×10^{1})	(3.48×10^{0})	(1.25×10^{-4})	(9.92×10^{-1})	(0)	(0)
Ackley	8.20×10^{-8}	2.33×10^{-7}	7.29×10^{-14}	3.66×10^{-7}	7.98×10^{-15}	4.00×10^{-15}	4.44×10^{-16}
	(6.73×10^{-8})	(7.19×10^{-8})	(3.00×10^{-14})	(7.57×10^{-8})	(2.03×10^{-15})	(0)	(0)
Griewank	1.53×10^{-3}	9.01×10^{-12}	9.75×10^{-3}	9.02×10^{-9}	4.83×10^{-3}	0	0
	(4.32×10^{-3})	(1.84×10^{-11})	(8.33×10^{-3})	(8.57×10^{-9})	(8.63×10^{-3})	(0)	(0)
Penalized 1	8.10×10^{-16}	1.96×10^{-15}	2.71×10^{-29}	6.45×10^{-14}	1.59×10^{-32}	4.32×10^{-1}	8.26×10^{-1}
	(1.07×10^{-15})	(1.11×10^{-15})	(1.88×10^{-28})	(3.70×10^{-14})	(1.03×10^{-33})	(6.20×10^{-2})	(7.31×10^{-2})
Penalized 2	3.26×10^{-13}	2.70×10^{-14}	2.79×10^{-28}	1.25×10^{-12}	4.39×10^{-4}	2.27×10^{0}	2.67×10^{0}
	(3.70×10^{-13})	(1.57×10^{-14})	(2.18×10^{-28})	(9.45×10^{-12})	(2.20×10^{-3})	(2.33×10^{-1})	(7.91×10^{-2})

TABLE 5.10

Comparisons between MPCA variants and ABCs on optimizing 30-dimensional functions. Results were taken from [24]. The shaded cells represent the best results overall.

Fun	Max.FEs	RABC	MABC	PABC	OGABC	BCABC	MPCA	QRMPCA
f_1 Sphere	150000	9.1×10^{-61}	9.43×10^{-32}	2.63×10^{-67}	6.35×10^{-60}	2.84×10^{-67}	4.79×10^{-139}	0
		(2.1×10^{-60})	(6.67×10^{-32})	(2.23×10^{-67})	(8.85×10^{-60})	(9.77×10^{-68})	(2.01×10^{-140})	(0)
f_9 Schwefel 2.22	200000	6.3×10^{-56}	2.40×10^{-17}	7.97×10^{-35}	3.58×10^{-41}	1.22×10^{-47}	1.41×10^{-68}	0
		(4.5×10^{-56})	(9.02×10^{-18})	(4.51×10^{-35})	(7.86×10^{-41})	(7.18×10^{-48})	(4.74×10^{-69})	(0)
f_{10} Rosenbrock	300000	4.3×10^{-1}	4.79×10^{-1}	3.85×10^{-3}	9.85×10^{-1}	2.28×10^{-2}	2.81×10^{1}	2.87×10^{1}
		(3.7×10^{-1})	(3.58×10^{-1})	(6.65×10^{-3})	(8.36×10^{-1})	(1.06×10^{-2})	(2.15×10^{-1})	(4.64×10^{-2})
f_6 Step	10000	5.3×10^{0}	6.13×10^{2}	1.48×10^{1}	9.83×10^{0}	4.33×10^{0}	0	0
		(2.8×10^{0})	(1.72×10^{2})	(7.78×10^{0})	(4.57×10^{0})	(1.73×10^{-1})	(0)	(0)
f_{11} Schwefel 2.26	100000	3.8×10^{-6}	6.35×10^{-16}	0	0	0	7.56×10^{3}	7.88×10^{3}
		(5.2×10^{-7})	(8.17×10^{-16})	(0)	(0)	(0)	(2.59×10^{2})	(3.40×10^{2})
f_{13} Rastrigin	100000	7.4×10^{-8}	0	0	0	0	0	0
		(2.6×10^{-8})	(0)	(0)	(0)	(0)	(0)	(0)
f_{19} Ackley	50000	9.6×10^{-7}	4.50×10^{-4}	6.07×10^{-8}	6.45×10^{-8}	2.97×10^{-10}	4.00×10^{-15}	4.44×10^{-16}
		(8.3×10^{-7})	(8.87×10^{-5})	(8.65×10^{-8})	(2.86×10^{-8})	(1.16×10^{-10})	(0)	(0)
f_{18} Griewank	50000	8.7×10^{-8}	8.36×10^{-8}	6.32×10^{-12}	8.36×10^{-10}	1.99×10^{-15}	0	0
		(2.1×10^{-8})	(2.71×10^{-8})	(1.41×10^{-11})	(1.54×10^{-10})	(2.40×10^{-15})	(0)	(0)
f_{23} Penalized 1	50000	5.4×10^{-16}	3.02×10^{-9}	6.46×10^{-21}	5.89×10^{-18}	2.21×10^{-21}	7.22×10^{-1}	1.03×10^{0}
		(2.8×10^{-16})	(3.01×10^{-9})	(4.92×10^{-21})	(6.85×10^{-19})	(7.28×10^{-21})	(5.32×10^{-2})	(8.23×10^{-2})
f_{24} Penalized 2	50000	1.5×10^{-12}	1.09×10^{-7}	3.35×10^{-20}	4.78×10^{-17}	6.60×10^{-21}	2.76×10^{0}	2.87×10^{0}
		(2.7×10^{-12})	(3.73×10^{-8})	(1.58×10^{-19})	(2.06×10^{-17})	(4.41×10^{-21})	(8.97×10^{-2})	(9.74×10^{-2})
f_{14} Alpine	300000	5.4×10^{-14}	6.58×10^{-15}	3.14×10^{-15}	6.35×10^{-17}	6.90×10^{-76}	4.16×10^{-72}	0
		(2.6×10^{-14})	(4.73×10^{-16})	(1.74×10^{-15})	(7.23×10^{-17})	(1.16×10^{-75})	(2.16×10^{-72})	(0)

References

[1] Ahandani, M. A. Opposition-based learning in the shuffled bidirectional differential evolution algorithm. *Swarm and Evolutionary Computation*, **26** (2016) 64–85.

[2] Al-Qunaieer, F. S., Tizhoosh, H. R., Rahnamayan, S. Opposition based computing–a survey. *2010 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2010.

[3] Aluffi-Pentini, F., Parisi, V., Zirilli, F. Global optimization and stochastic differential equations. *Journal of optimization theory and applications*, **47(1)** (1985) 1–16.

[4] Anochi, J. A. and Campos Velho, H. F. Optimization of feedforward neural network by Multiple Particle Collision Algorithm. *2014 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, IEEE, 2014 (128–134).

[5] Anochi, J. A., Campos Velho, H. F., Furtado, H. C. M., Luz, E. F. P. Self-configuring Two Types of Neural Networks by MPCA. *Journal of Mechanics Engineering and Automation*, **5** (2015) 112–120.

[6] Antoniou, A., Lu, W.-S. *Practical optimization: algorithms and engineering applications*. Springer Science & Business Media, 2007.

[7] Beyer, H.-G. *The theory of evolution strategies*. Springer Science & Business Media, 2013.

[8] Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, **10(6)** (2006) 646–657.

[9] Chelouah, R., Siarry, P. A continuous genetic algorithm designed for the global optimization of multimodal functions. *Journal of Heuristics* **6(2)** (2000) 191–213.

[10] Das, S., Biswas, A., Dasgupta, S., Abraham, A. Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. *Foundations of Computational Intelligence*, Springer **3**, (2009) 23–55.

[11] Das, S., Suganthan, P. N. Differential evolution: a survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, **15(1)** (2011) 4–31.

[12] De Castro, L. N., Timmis, J. Artificial immune systems: a new computational intelligence approach. Springer Science & Business Media, 2002.

[13] Mendiburu, F., Simon, R. Agricolae-ten years of an open source statistical tool for experiments in breeding, agriculture and biology. Technical report, PeerJ PrePrints, 2015.

[14] Dorigo, M. and Birattari, M. Ant colony optimization. *Encyclopedia of machine learning*, Springer (2010) 36–39.

[15] Dorigo, M., Birattari, M., Stützle, T. Ant colony optimization. *Computational Intelligence Magazine*, IEEE, **1(4)** (2006) 28–39.

[16] Du, K.-L., Swamy, M. N. S. *Search and Optimization by Metaheuristics*. Springer, 2016.

[17] Eberhart, R. C., Kennedy, J. A new optimizer using particle swarm theory. *Proceedings of the sixth international symposium on micro machine and human science*, New York, **1** (1995) 39–43.

[18] Echevarría, L. C., Llanes Santiago, O., Silva Neto, A. J. Aplicación de los algoritmos Evolución Diferencial y Colisión de Partículas al diagnóstico de fallos en sistemas industriales. *Revista Investigación Operacional*, **33(2)** (2012) 160–172.

[19] Ergezer, M., Simon, D., Du, D. Oppositional biogeography-based optimization. *IEEE International Conference on Systems, Man and Cybernetics (SMC 2009).*, IEEE, (2009) 1009–1014.

[20] Fogel, L. J. *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., New York, 1999.

[21] Gao, W.-F., Liu, S.-Y. A modified artificial bee colony algorithm. *Computers & Operations Research*, **39(3)** (2012) 687–697.

[22] Gao, W.-F., Liu, S.-Y., Huang, L.-L. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Transactions on Cybernetics*, **43(3)** (2013) 1011–1024.

[23] Gao, W.-F., Liu, S.-Y., Huang, L.-L. A novel artificial bee colony algorithm with powell's method. *Applied Soft Computing*, **13(9)** (2013) 3763–3775.

[24] Gao, W., Chan, F. T. S., Huang, L. L., Liu, S. Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood. *Information Sciences*, **316** (2015) 180–200.

[25] Geem, Z. W., Kim, J. H., Loganathan, G. V. A new heuristic optimization algorithm: harmony search. *Simulation*, **76(2)** (2001) 60–68.

[26] Guo, Z., Wang, S., Yue, X., Yang,H. Global harmony search with generalized opposition-based learning. *Soft Computing*, **21(8)** (2017) 2129–2137.

[27] Guo, Z., Yue, X., Zhang, K., Deng, C., Liu, S. Enhanced social emotional optimisation algorithm with generalised opposition–based learning. *International Journal of Computing Science and Mathematics*, **6(1)** (2015) 59–68.

[28] Han, L., He, X. A novel opposition-based particle swarm optimization for noisy problems. *Third International Conference on Natural Computation (ICNC 2007)*, IEEE, **3** (2007) 624–629.

[29] Holland, J. H. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992.

[30] Jamil, M., Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4(2)** (2013) 150–194.

[31] Fister Jr., I., Yang, X.-S., Fister, I., Brest, J., Fister, D. *A brief review of nature-inspired algorithms for optimization.* CoRR, abs/1307.4186, 2013.

[32] Kalra, S., Sriram, A., Rahnamayan, S., Tizhoosh, H. R. *Learning opposites using neural networks.* CoRR, abs/1609.05123, 2016.

[33] Kang, F., Li, J., Ma, Z. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, **181(16)** (2011) 3508–3531.

[34] Karaboga, D. *An idea based on honey bee swarm for numerical optimization.* Technical report, Technical report-tr06, Erciyes university, Engineering faculty, Computer Engineering Department, 2005.

[35] Karaboga, D., Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization*, **39(3)** (2007) 459–471.

[36] Kennedy, J. Particle swarm optimization. *Encyclopedia of Machine Learning*, Springer (2010) 760–766

[37] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. Optimization by simulated annealing. *Science* **220(4598)** (1983) 671–680.

[38] Kuo, R. J., Zulvia, F. E. The gradient evolution algorithm: A new metaheuristic. *Information Sciences*, **316** (2015) 246–265.

[39] Langdon, W. B., Gustafson, S. M. Genetic programming and evolvable machines: ten years of reviews. *Genetic Programming and Evolvable Machines Tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines*, **11(3/4)** (2010) 321–338.

[40] Liang, J. J., Qin, A. K., Suganthan, P. N., Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation*, **10(3)** (2006) 281–295.

[41] Liang, J. J., Qu, B. Y., Suganthan, P. N. *Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization.* Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2013.

[42] Lin, M.-H., Tsai, J.-F., Yu, C.-S. A review of deterministic optimization methods in engineering and management. *Mathematical Problems in Engineering*, 2012.

[43] Luz, E. F. P., Becceneri, J. C., Campos Velho, H. F. A new multi-particle collision algorithm for optimization in a high performance environment. *Journal of Computational Interdisciplinary Sciences*, **1(1)** (2008) 3–10.

[44] Malisia, A. R., Tizhoosh, H. R. Applying opposition-based ideas to the ant colony system. *Swarm Intelligence Symposium (SIS 2007)*. IEEE. (2007) 182–189.

[45] Mendes, R., Kennedy, J., Neves, J. The fully informed particle swarm: simpler, maybe better. *IEEE transactions on evolutionary computation*, **8(3)** (2004) 204–210.

[46] Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, 2016.

[47] Molga, M., Smutnicki, C. *Test functions for optimization needs*. 2005.

[48] Moscato, P. On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms, 1989.

[49] Park, S.-Y., Lee, J.-J. Stochastic opposition-based learning using a beta distribution in differential evolution. *IEEE transactions on cybernetics*, **46(10)** (2016) 2184–2194.

[50] Pohlheim, H. Examples of objective functions. *Retrieved*, **4(10)** (2007).

[51] Price, K., Storn, R. M., Lampinen, J. A. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[52] Qin, A. K., Huang, V. L., Suganthan, P. N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, **13(2)** (2009) 398–417.

[53] Rahnamayan, S., Tizhoosh, H. R., Salama, M. M. A. Opposition-based differential evolution for optimization of noisy problems. *IEEE Congress on Evolutionary Computation, (CEC 2006)*, IEEE, (2006) 1865–1872.

[54] Rahnamayan, S., Tizhoosh, H. R., Salama, M. Quasi-oppositional differential evolution. *IEEE Congress on Evolutionary Computation (CEC 2007)*, IEEE (2007) 2229–2236

[55] Ratnaweera, A., Halgamuge, S. K., Watson, H. C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on evolutionary computation*, **8(3)** (2004) 240–255.

[56] Reynolds, R. G. An introduction to cultural algorithms. *Proceedings of the third annual conference on evolutionary programming*, Singapore. (1994) 131–139.

[57] Sacco, W. F., Oliveira, C. R. E. A new stochastic optimization algorithm based on a particle collision metaheuristic. *Proceedings of 6th World Congress of Structural and Multidisciplinary Optimisation (WCSMO)*, 2005.

[58] Sacco, W. F., Pereira, C. M. N. A. Two stochastic optimization algorithms applied to nuclear reactor core design. *Progress in Nuclear Energy*, **48(6)** (2006) 525–539.

[59] Sambatti, S. B. M., Anochi, J. A., Luz, E. F. P., Carvalho, A. R., Shiguemori, E. H., Campos Velho, H. F. Automatic configuration for neural network applied to atmospheric temperature profile identification. *3rd International Conference on Engineering Optimization*, (2012) 1–9.

[60] Shah-Hosseini, H. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation*, **1(1-2)** (2009) 71–79.

[61] Shi, Y., Eberhart, R. C. Empirical study of particle swarm optimization. *Proceedings of the Congress on Evolutionary Computation (CEC 99)*, IEEE. **3** (1999).

[62] Simon, D. Biogeography-based optimization. *IEEE transactions on evolutionary computation*, **12(6)** (2008) 702–713.

[63] Storn, R., Price, K. *Differential evolution – A simple efficient adaptive scheme for global optimization over continuous spaces*. Technical Report 95-012, Int. Compt. Sci. Inst., Berkeley, CA, 1995.

[64] Sulimov, V. D., Shkapov, P. M. Application of hybrid algorithms to computational diagnostic problems for hydromechanical systems. *Journal of Mechanics Engineering and Automation*, **2(12)** (2012) 734–741.

[65] Tizhoosh, H. R. Opposition-Based Learning: A New Scheme for Machine Intelligence. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, IEEE, **1** (2005) 695–701.

[66] Tizhoosh, H. R., Ventresca, M., Rahnamayan, S. Opposition-Based Computing, *Oppositional Concepts in Computational Intelligence*, Springer Berlin Heidelberg (2008) 11–28.

[67] Tizhoosh, H. R., Rahnamayan, S. Learning opposites with evolving rules. *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2015.

[68] Tizhoosh, H. R., Ventresca, M. Oppositional Concepts in Computational Intelligence. *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2008.

[69] Ventresca, M., Tizhoosh, H. R. Improving the convergence of backpropagation by opposite transfer functions. *International Joint Conference on Neural Networks (IJCNN'06)*, IEEE, (2006) 4777–4784.

[70] Vesterstrøm, J., Thomsen, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. *Congress on Evolutionary Computation (CEC2004)*, **2** (2004) 1980–1987.

[71] Wang, H., Li, H., Liu, Y., Li, C. and Zeng,S. Opposition-based particle swarm algorithm with Cauchy mutation. *IEEE Congress on Evolutionary Computation (CEC 2007)*, IEEE (2007) 4750–4756.

[72] Xu, Q., Wang, L., Wang, N., Hei, X., Zhao, L. A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence*, **29** (2014) 1–12.

[73] Yang, X.-S.. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[74] Zhan, Z.-H., Zhang, J., Li, Y., Shi, Y.-H. Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, **15(6)** (2011) 832–847.

[75] Zhang, J., Sanderson, A. C. JADE: adaptive differential evolution with optional external archive. *IEEE transactions on evolutionary computation*, **13(5)** (2009) 945–958.

A. Benchmark functions. The A.1 gives the information of the benchmark optimization functions used to compare the optimization algorithms presented in the paper.

TABLE A.1
List of benchmark functions and their properties

Function name	Definition ($f(x^*) = \min_{x \in S} f(x_1, \cdots, x_D)$)	Search space/$f(x^*)$
Sphere	$f_1(x) = \sum_{d=1}^{D} x_d^2$	$\pm 100/0$
Powell sum	$f_2(x) = \sum_{d=1}^{D} \lvert x_d \rvert^{(d+1)}$	$\pm 100/0$
Sum squares	$f_3(x) = \sum_{d=1}^{D} d x_d^2$	$\pm 100/0$
Quartic w/ noise	$f_4(x) = \sum_{d=1}^{D} d x_d^4$ + random[0,1)	$\pm 1.28/0$
Schwefel 2.21	$f_5(x) = \max_{1 \le d \le D} \lvert x_d \rvert$	$\pm 100/0$
Step	$f_6(x) = \sum_{d=1}^{D} \lfloor x_d + 0.5 \rfloor^2$	$\pm 100/0$
Schwefel 1.2	$f_7(x) = \sum_{d=1}^{D} \left(\sum_{j=1}^{d} x_j \right)^2$	$\pm 100/0$
Dixon & Price	$f_8(x) = (x_1 - 1)^2 + \sum_{d=2}^{D} d(2x_d^2 - x_{d-1})^2 - 1$	$\pm 10/0$
Schwefel 2.22	$f_9(x) = \sum_{d=1}^{D} \lvert x_d \rvert + \prod_{d=1}^{D} \lvert x_d \rvert$	$\pm 10/0$
Rosenbrock	$f_{10}(x) = \sum_{d=1}^{D-1} (100(x_d^2 - x_{d+1})^2 + (1 - x_d)^2)$	$\pm 30/0$
Schwefel 2.26	$f_{11}(x) = \sum_{d=1}^{D} \left(-x_d sin \left(\sqrt{\lvert x_d \rvert} \right) \right) + \alpha \cdot D$	$\pm 500/0$
	$\alpha = 418.98288727243379980791360 1398$	
Michalewicz	$f_{12}(x) = -\sum_{d=1}^{D} \sin(x_d) \sin(d x_d^2/\pi)^{20}$	$\pm \pi/0$
Rastrigin	$f_{13}(x) = 10 \cdot D + \sum_{d=1}^{D} (x_d^2 - 10 \cos(2\pi x_d))$	$\pm 5.12/0$
Alpine 1	$f_{14}(x) = \sum_{d=1}^{D} \lvert x_d \sin x_d + 0.1 x_d \rvert$	$\pm 100/0$
Levy	$f_{15}(x) = \sin^2(\pi y_1) + (y_D - 1)^2 \left(1 + \sin^2(2\pi y_D)\right)$	$\pm 10/0$
	$+ \sum_{d=1}^{D-1} (y_d - 1)^2 \left(1 + 10 \sin^2 \left(\pi y_{d+1}\right)\right), y_d = 1 + \frac{x_d - 1}{4}$	
Exponential	$f_{16}(x) = \exp\left(-0.5 \sum_{d=1}^{D} x_d^2\right)$	$\pm 100/0$
Rana	$f_{17}(x) = \sum_{d=1}^{D-1} x_d \sin m_d \cos p_d + (x_{d+1} + 1) \cos m_d \sin p_d$	$\pm 500/0$
	$m_d = \sqrt{\lvert x_{d+1} + 1 - x_d \rvert}, p_d = \sqrt{\lvert x_{d+1} + 1 + x_d \rvert}$	
Griewank	$f_{18}(x) = 1 + \frac{1}{4000} \sum_{d=1}^{D} x_d^2 - \prod_{d=1}^{D} \cos\left(\frac{x_d}{\sqrt{d}}\right)$	$\pm 600/0$
Ackley	$f_{19}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{d=1}^{D} x_d^2}\right)$	$\pm 32/0$
	$- \exp\left(\frac{1}{D} \sum_{d=1}^{D} \cos\left(2\pi x_d\right)\right) + 20 + e$	
Zakharov	$f_{20}(x) = \sum_{d=1}^{D} x_d^2 + \left(\sum_{d=1}^{D} 0.5 d x_d\right)^2 + \left(\sum_{d=1}^{D} 0.5 d x_d\right)^4$	$[-5, 10]/0$
Salomon	$f_{21}(x) = 1 - \cos(2\pi \lvert\lvert x \rvert\rvert) + 0.1 \lvert\lvert x \rvert\rvert, \lvert\lvert x \rvert\rvert = \sqrt{\sum_{d=1}^{D} x_d^2}$	$\pm 100/0$
Egg Holder	$f_{22}(x) = \sum_{d=1}^{D-1} -\left(x_{d+1} + 47\right) \sin \sqrt{\lvert x_{d+1} + 0.5 x_d + 47 \rvert}$	$\pm 512/V$
	$- x_d \sin \sqrt{\lvert x_d - x_{d+1} - 47 \rvert}$	
Penalized 1	$f_{23}(x) = \frac{\pi}{n} \left(10 \sin^2(\pi y_1) + (y_n - 1)^2\right)$	$\pm 50/0$
	$+ \frac{\pi}{n} \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + 10 \sin^2(\pi y_{i+1})\right]$	
	$+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	
	$y_i = 1 + \frac{1}{4}(x_i + 1),$	
	$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$	
Penalized 2	$f_{24}(x) = 0.1 \left(\sin^2(3\pi x_1) + (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n)\right]\right)$	$\pm 50/0$
	$+ 0.1 \sum_{i=1}^{n-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right]$	
	$+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	
Rotated Rastrigin	$f_{25}(x) = f_{13}(z), z = x * M$	$\pm 5.12/0$
Rotated Griewank	$f_{26}(x) = f_{18}(z), z = x * M$	$\pm 600/0$
Rotated Ackley	$f_{27}(x) = f_{19}(z), z = x * M$	$\pm 32/0$
Shifted Rastrigin	$f_{28}(x) = f_{13}(z), z = x - o$	$\pm 5.12/0$
Shifted Griewank	$f_{29}(x) = f_{18}(z), z = x - o$	$\pm 600/0$
Shifted Ackley	$f_{30}(x) = f_{19}(z), z = x - o$	$\pm 32/0$