Fuzzy Dynamic Adaptation of Parameters in the Water Cycle Algorithm

Eduardo Méndez, Oscar Castillo, José Soria and Ali Sadollah

Abstract This paper describes the enhancement of the water cycle algorithm (WCA) using a fuzzy inference system to dynamically adapt its parameters. The original WCA is compared in terms of performance with the proposed method called WCA with dynamic parameter adaptation (WCA-DPA). Simulation results on a set of well-known test functions show that the WCA is improved with a fuzzy dynamic adaptation of the parameters.

Keywords WCA · Fuzzy logic · Optimization

1 Introduction

Dynamic parameter adaptation can be performed in many ways, the most common approaches being linearly increasing or decreasing a parameter, and other approaches include nonlinear or stochastic functions. In this paper, a different approach is taken, which is using a fuzzy inference system (FIS) to replace a function or to change its behavior, with the final purpose of improving the performance of the water cycle algorithm (WCA). The WCA is a population-based and nature-inspired metaheuristic, which is inspired on a simplified form of the water cycle process [2, 12].

Using a FIS to enhance global-optimization algorithms is an active area of research; some works of enhancing particle swarm optimization are PSO-DPA [9], APSO [17] and FAPSO [13]. Since the WCA has some similarities with PSO, a FIS similar to the one in [9] was developed.

E. Méndez · O. Castillo (✉) · J. Soria
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: ocastillo@tectijuana.mx

A. Sadollah
Nanyang Technological University, Singapore, Singapore
e-mail: ali_sadollah@yahoo.com

P. Melin et al. (eds.), *Nature-Inspired Design of Hybrid Intelligent Systems*,
Studies in Computational Intelligence 667, DOI 10.1007/978-3-319-47054-2_20

A comparative study was conducted which highlights the similarities and differences with other hierarchy-based metaheuristics. In addition, a performance study between the proposed water cycle algorithm with Dynamic Parameter Adaptation (WCA-DPA) and the original WCA was also conducted, using ten well-known test functions frequently used in tin the literature.

This paper is organized as follows. In Sect. 2 the WCA is described. In Sect. 3, some similarities with other metaheuristics are highlighted. Section 4 is about how to improve the WCA with fuzzy parameter adaptation. A comparative study is also presented in Sect. 5 and, finally in Sect. 6 some conclusions and future work are given.

2 Nonlinear the Water Cycle Algorithm

The WCA is a population-based and nature-inspired metaheuristic, where a population of streams is formed from rainwater drops. This population of streams follows a behavior inspired on the hydrological cycle. In which streams flows downhill, then they form rivers, which also flow downhill towards the sea. This process in which streams flows toward rivers and rivers towards the sea is a simplified form of the *runoff* process of the hydrologic cycle. Some of those streams are evaporated and some new streams are formed from rain as part of the hydro-logic cycle.

2.1 The Landscape

There are a number of landforms involved in the hydrologic cycle, for example: streams, rivers, lakes, valleys, mountains, and glaciers. But in the WCA only three of them are considered, which are streams, rivers, and seas, and in fact there is only one sea. In this subsection the structure, preprocessing and initialization of the algorithm are described.

In the WCA an individual (a.k.a. stream), is an object which consist of n variables grouped as a n-dimensional column vector

$$\mathbf{x}_k = [x_{k1}, \ldots, x_{kn}]^{\mathrm{T}} \in \mathbb{R}^n. \tag{1}$$

And the whole population of N streams is denoted by

$$\mathbf{X} = \{\mathbf{x}_k | k = 1, 2, \ldots, N\}, \tag{2}$$

which is often represented as a $N \times n$ matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nn} \end{bmatrix}. \tag{3}$$

In short each row is an individual stream and their columns are their variables. From the whole population some of those streams will become rivers and another one will become the sea. The number of streams and rivers are defined by the following equations:

$$N_{\text{sr}} = \text{Number of Rivers} + \underbrace{1}_{\text{sea}}, \tag{4}$$

$$N_{\text{streams}} = N - N_{\text{sr}}, \tag{5}$$

where N_{sr} is a value established as a parameter of the algorithm and N_{streams} is the number of remaining streams. Which individuals become rivers or sea will depend on the fitness of each stream. To obtain the fitness, first we need an initial population matrix X, and this matrix is initialized with random values as follows:

$$\mathbf{x}_i = \mathbf{b}_{\text{lower}} + \mathbf{r} \cdot (\mathbf{b}_{\text{lower}} - \mathbf{b}_{\text{upper}}), \quad \text{for} \quad i = 1, 2, \ldots, N, \tag{6}$$

where $\mathbf{b}_{\text{lower}}$, $\mathbf{b}_{\text{upper}}$ are vectors in $\mathbb{R}^n$ with the lower and upper bounds for each dimension which establish the search-space, and $\mathbf{r}$ is an n-dimensional vector of independent and identically distributed (i.i.d) values, that follows a uniform distribution:

$$\mathbf{r} \sim U(0, 1)^n. \tag{7}$$

Once an initial population is created the fitness of each stream $\mathbf{x}_i$ is obtained by:

$$\mathbf{f}_i = f(\mathbf{x}_i) = f(x_{i1}, x_{i2}, \ldots x_{in}), \quad \text{for} \quad i = 1, 2, 3, \ldots, N, \tag{8}$$

where $f(\cdot)$ is a problem dependent function to estimate the fitness of a given stream. This fitness function it is what the algorithm tries to optimize.

Sorting the individuals by fitness and in ascending order using:

$$[\mathbf{X}, \mathbf{f}] \leftarrow \text{sort}](\mathbf{X}, \mathbf{f}), \tag{9}$$

the first individual becomes the sea, the next $N_{\text{sr}-1}$ the rivers, and the following N_{streams} individuals turn into the streams who flow toward the rivers or sea, as show in:

$$\mathbf{X} = \begin{matrix} sea\{ \\ rivers\{ \\ \\ \\ streams\{ \\ \\ \\ \end{matrix} \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_{N_{sr}}^\top \\ \mathbf{x}_{N_{sr}+1}^\top \\ \vdots \\ \mathbf{x}_{N_{sr}+N_{streams}}^\top \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nn} \end{bmatrix}. \tag{10}$$

Each of the N_{streams} is assigned to a river or sea, this assignment can be done randomly. But the *stream order* [14, 15], which is the number of streams assigned to each river/sea is calculated by:

$$\mathbf{so}_i \left\lfloor \left| \frac{\mathbf{f}_i}{\sum_{j=1}^{N_{\text{sr}}} \mathbf{f}_j + \varepsilon} \right| \cdot N_{\text{streams}} \right\rfloor, \quad n = 1, 2, \ldots, N_{\text{sr}}, \tag{11}$$

$$\mathbf{so}_1 \leftarrow \mathbf{so}_1 + \left(N_{\text{streams}} - \sum_{i=1}^{N_{\text{sr}}} \mathbf{so}_i \right), \tag{12}$$

where $\varepsilon \approx 0$. The idea behind the Eq. (11) is that the amount of water (streams) entering a river or sea varies so when a river is more abundant (has a better fitness) than another, it means that more streams flow into the river, hence the discharge (stream-flow) is higher. This means, the streamflow magnitude of rivers is inversely proportional to its fitness in the case of minimization problems.

The Eq. (11) has been changed from the original proposed in [2], the round function was replaced by a floor function, a value of ε was added to the divisor, and Eq. (12) was also added to handle the remaining streams. These changes are for the implementation purposes and an alternative to the method proposed in [12].

After obtaining the stream order of each river and sea, the streams are randomly distributed between them.

2.2 The Run-off Process

The run-off process is one of the three processes considered in the WCA, which handles the way water flows in form of streams and rivers towards the sea. The following equations describe how the flow of streams and rivers are simulated at a given instant (iteration):

$$\mathbf{x}_{\text{stream}}^{i+1} = \mathbf{x}_{\text{stream}}^{i} + \mathbf{r} \cdot C \cdot (\mathbf{x}_{\text{sea}}^{i} - \mathbf{x}_{\text{stream}}^{i}), \tag{13}$$

$$\mathbf{x}_{\text{stream}}^{i+1} = \mathbf{x}_{\text{stream}}^{i} + \mathbf{r} \cdot C \cdot (\mathbf{x}_{\text{river}}^{i} - \mathbf{x}_{\text{stream}}^{i}), \tag{14}$$

$$\mathbf{x}_{\text{river}}^{i+1} = \mathbf{x}_{\text{river}}^{i} + \mathbf{r} \cdot C \cdot (\mathbf{x}_{sea}^{i} - \mathbf{x}_{\text{river}}^{i}), \tag{15}$$

for $i = 1, 2, \ldots, N_{it}$, where N_{it} and C are parameters of the algorithm, and $\mathbf{r}$ is a vector with i.i.d values defined by Eq. (7), although any other distribution could be used.

The Eq. (13) defines the movement of streams who flow directly to the sea, Eq. (14) is for streams who flow toward the rivers, and Eq. (15) is for the rivers flow toward the sea. A value of $C > 1$ enables streams to flow in different directions toward the rivers or sea. Typically, the value of C is chosen from the range (1,2] being 2 the most common.

2.3 Evaporation and Precipitation Processes

The runoff process of the WCA basically consists of moving indirectly toward the global best (sea). Algorithms focused on following the global best although they are really fast, tend to premature convergence or stagnation. The way in which WCA deals with exploration and convergence is with the evaporation and precipitation processes. So when streams and rivers are close enough to the sea, some of those streams are evaporated (discarded) and then new streams are created as part of the precipitation process. This type of reinitialization is similar to the cooling down and heating up reinitialization process of the simulated annealing algorithm [3].

The evaporation criterion is: if a river is close enough to the sea, then the streams assigned to that river are evaporated (discarded) and new streams are created by raining around the search space. To evaporate the streams of a given river the following condition must be satisfied:

$$\underbrace{|\mathbf{x}_{\text{sea}} - \mathbf{x}_{\text{river}}| < d_{\max}}_{\text{evaporation criterion}}, \tag{16}$$

where $d_{\max} \approx 0$ is a parameter of the algorithm. This condition must be applied to every river, and if its satisfied each stream who flow toward this river must be replaced as:

$$\underbrace{\mathbf{x}_{\text{stream}} = \mathbf{b}_{\text{lower}} + \mathbf{r} \cdot (\mathbf{b}_{\text{lower}} - \mathbf{b}_{\text{upper}})}_{\text{raining around the search space}}, \tag{17}$$

a high value of $d_{\max}$ will favor the exploration and a low one will favor the exploitation.

To increase the exploration and exploitation around the sea an especial evaporation criterion is used for the streams, which flow directly to sea:

$$\underbrace{|\mathbf{x}_{\text{sea}} - \mathbf{x}_{\text{seastream}}| < d_{\max}}_{\text{evaporation criterion}}, \tag{18}$$

where $\mathbf{x}_{\text{seastream}}$ is a stream which flows directly to the sea. If this criterion defined by the inequality (18) is satisfied then, the stream is evaporated and a new one is created using:

$$\underbrace{\mathbf{x}_{\text{seastream}} = \mathbf{x}_{\text{sea}} + \mathbf{g}}_{\text{raining around the sea}}, \tag{19}$$

where $\mathbf{g}$ is an n-dimensional vector of independent and identically distributed (i.i.d) values, that follow a normal distribution:

$$\mathbf{g} \sim \mathcal{N}(\mu = 0, \sigma^2 = 0.01)^n. \tag{20}$$

2.4 Steps of WCA

The steps of WCA are summarized as an algorithm in Fig. 1.

Algorithm: Water Cycle Algorithm

input : $\mathrm{f}, N_{sr}, d_{\max}, N, \mathbf{b}_{lower}, \mathbf{b}_{upper}, it_{\max}$

1: Generate the initial population of streams (raindrops) using eq. (3).
2: Calculate the fitness of each stream by Eq. (8).
3: Sort the population of streams by fitness to determine streams, rivers and sea, as in (10).
4: Calculate the intensity of flow for rivers and sea using (11) and designate which streams flows towards each river or sea.
5: Streams flow towards the sea and rivers by Eqs. (13) and (14) respectively, and rivers flow to the sea by Eq. (15).
6: Update the fitness of each stream by Eq. (8). After each update check:
If a stream new fitness is better than his assigned river/sea, exchange positions.
If a river new fitness is better than the sea, exchange positions.
7: Check the evaporation conditions for both rivers and streams, using Eqs. (16) and (18) respectively, and start the evaporation and raining processes using Eqs. (17) and (19).
8: Check the convergence criterion. If satisfied stop the algorithm, otherwise return to the step 5 .

output: The individual with the best fitness also known as the sea.

Fig. 1 The water cycle algorithm (WCA)

3 Similarities and Differences with Other Metaheuristics

The WCA has some similarities with other metaheuristics, but yet is different from those. Some of the similarities and differences have already been studied, for example: In [2], differences and similarities with particle swarm optimization (PSO) [7] and genetic algorithms (GA) [4] are explained. In [11], WCA is compared with the Imperialist Competitive Algorithm (ICA) [1] and PSO [7]. In [12], the WCA is compared with two nature-inspired metaheuristics: the intelligent water drops [5] and water wave optimization [18]. So far similarities and differences with population-based and nature-inspired metaheuristics have been studied, in this subsection, WCA is compared with two metaheuristics who use a hierarchy.

3.1 Hierarchical Particle Swarm Optimization

In hierarchical particle swarm optimization (H-PSO), particles are arranged in a regular tree hierarchy that defines the neighborhoods structure [6]. This hierarchy is defined by a height h and a branching degree d, this is similar to the landscape (hierarchy) of the WCA, in fact the WCA would be like a tree of height $h = 3$ (sea, rivers, streams), but with varying branching degrees, since the level-2 consist of N_{sr} branches (rivers) and the level-3 depends of the stream orders (**so**), so WCA hierarchy it is not a nearly regular tree like in the H-PSO.

Another difference is that H-PSO uses velocities to update the positions of the particles just like in standard PSO. But a similarity is that instead of moving towards the global best like in PSO they move toward their parent node, just like streams flow to rivers and rivers flow to the sea. As in WCA, in H-PSO particles move up and down the hierarchy, and if a particle at a child node has found a solution that is better than the best so far solution of the particle at the parent node, the two particles are exchanged. This is similar yet different to the runoff process of the WCA, the difference being that WCA uses only the social component to update the positions, and H-PSO uses both the social and cognitive components, and also the velocity with inertia weight. The cognitive component and the inertia weight are the ways in which H-PSO deals with exploration and exploitation. The WCA uses the evaporation and precipitation processes for those phases.

3.2 Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Four types of grey wolves are simulated: alpha, beta, delta, and omega, which are employed for simulating the leadership hierarchy [10]. This social hierarchy is similar to the WCA hierarchy

with a $N_{sr} = 3$, where the alpha could be seen as the sea, the beta and delta as the rivers and the omegas as the streams. Although the hierarchy is similar, the way in which the GWO algorithm updates the positions of the individuals is different. GWO position update depends of the hunting phases: searching for prey, encircling prey, and attacking prey. Those hunting phases are the way in which the GWO deals with exploration and exploitation. As mentioned before, the WCA uses the evaporation and precipitation process, which are very different to the hunting phases.

4 Fuzzy Parameter Adaptation

The objective of dynamic parameter adaptation is to improve the performance of an algorithm by adapting its parameters. Dynamic parameter adaptation can be done in many ways, the most common being linearly increasing or decreasing a parameter, usually the acceleration coefficients. Other approaches include using nonlinear or stochastic functions. In this paper, a different approach is taken, which is using a fuzzy system to replace a function or to change its behavior.

In the WCA there are two parameters which can be adapted dynamically, that is while the algorithm is running. One is the parameter *C,* which is used in Eqs. (13)–(15) for updating the positions of the streams. The other one is the parameter d_{max} used in Eq. (16) as a threshold for the evaporation criterion. In [12] the parameter d_{max} is linearly decreased and a stochastic evaporation rate for every river is introduced, together both changes improve the performance of the WCA.

Since there are already improvements with the parameter d_{max}, the subject of study in this paper is the *C* parameter.

4.1 Mamdani's Fuzzy Inference System

A single-input and multiple-output (SIMO) Mamdani's FIS [8] was developed. The system consists of the *Iteration* input and the outputs *C* and C_{rivers}. In Fig. 2 the layout of the FIS is shown. The idea is to use different values of the parameter *C*, one for Eqs. (13) and (14) which are for the flow of streams and another one (C_{rivers}) for Eq. (15) which is for the flow of rivers.

Before going into the FIS, the Iteration input is scaled to the interval [0, 1] by the following equation:

$$\text{Iteration} = \frac{i}{N_{it}}, \quad \text{for} \quad i = 1, 2, \ldots, N_{it}. \tag{21}$$

The membership functions for the iteration input are shown in Fig. 3. The range of the first output *C* had been chosen as the interval (1.8, 3.7), and for the output

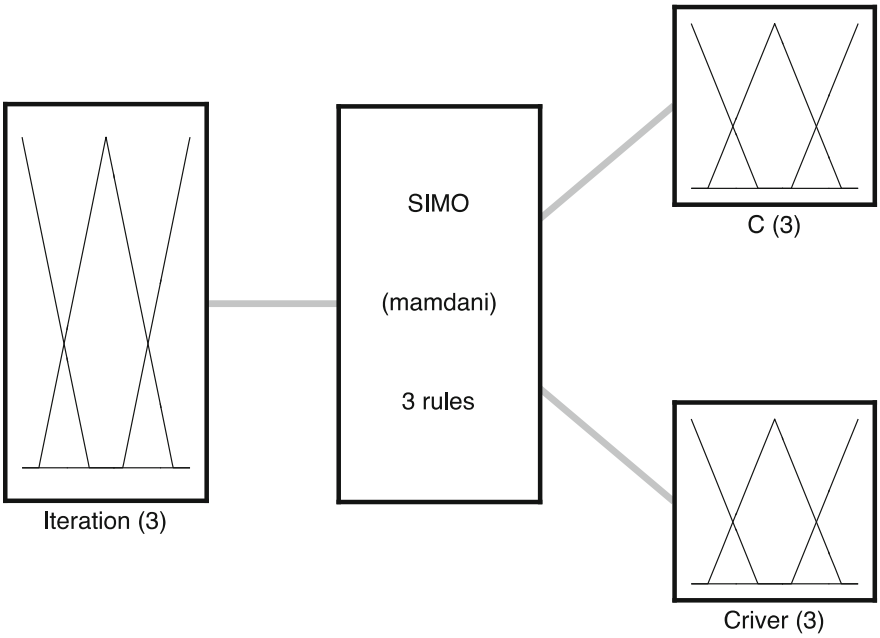

Fig. 2 Single-input and multiple-output Mamdani's fuzzy inference system

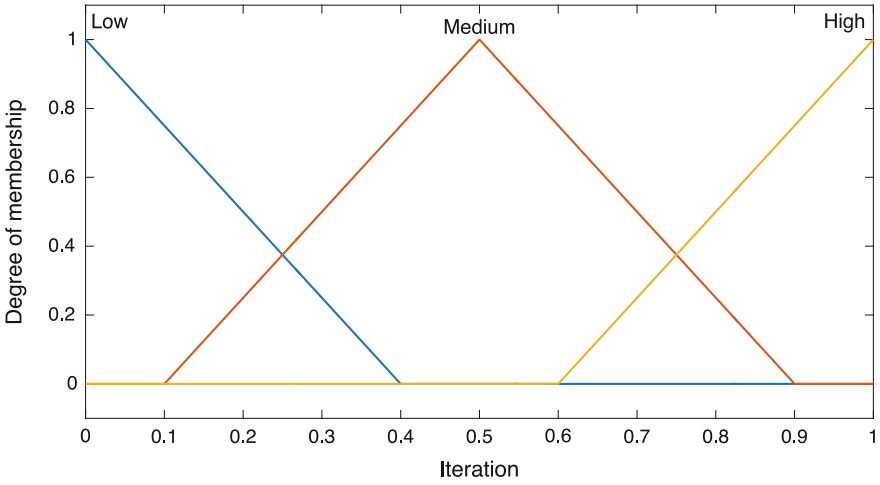

Fig. 3 Input: iteration

C_{rivers} the interval (2, 10), the details of the membership functions are shown in Figs. 4 and 5. The idea of using higher values for these parameters is to favor the exploration in the runoff process at an early stage. Since having a greater value than

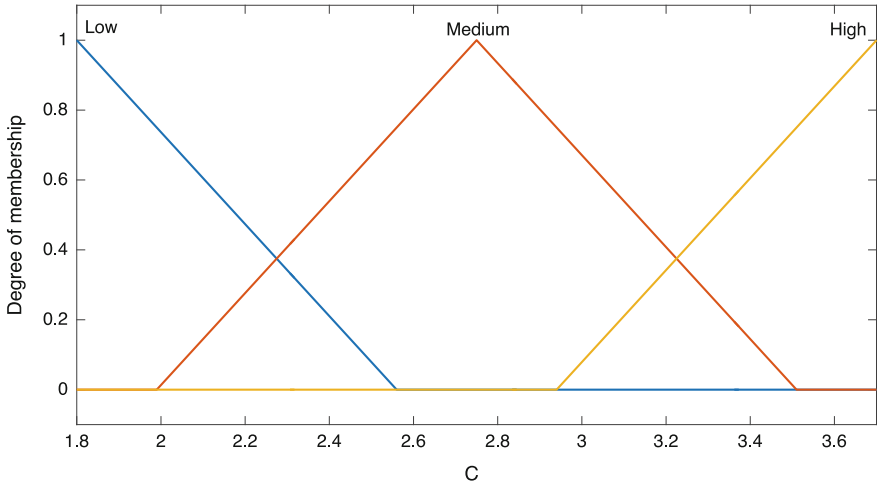

Fig. 4 Output: C

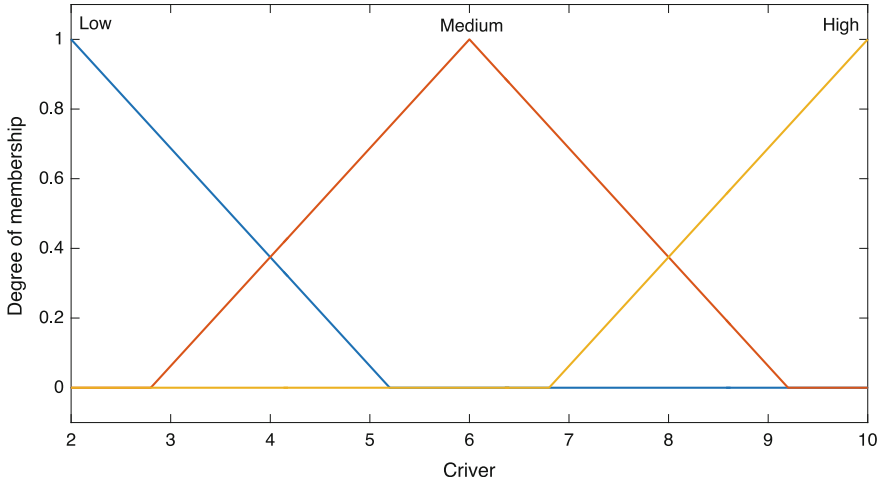

Fig. 5 Output: C_{rivers}

2 means that there is a higher probability of moving beyond the rivers or sea. For the special case of C_{rivers} it also helps to prevent the evaporation and precipitation processes. Figure 6 shows a flowchart of the WCA with the SIMO-FIS integrated.

5 Experiments and Comparisons

For the comparison between the WCA and the WCA-DPA, a subset of test functions was used. Although in the literature there is no agreed set of test functions for measuring the performance of algorithms, a diverse subset of 10 test functions who had been used before for some bio-inspired algorithms was chosen. In Table 1 the specifications of those functions are summarized and in Fig. 7 the plots for 2-dimensions viewed from a 3D perspective and from above are shown. In [16] there is a more detailed description of those functions.

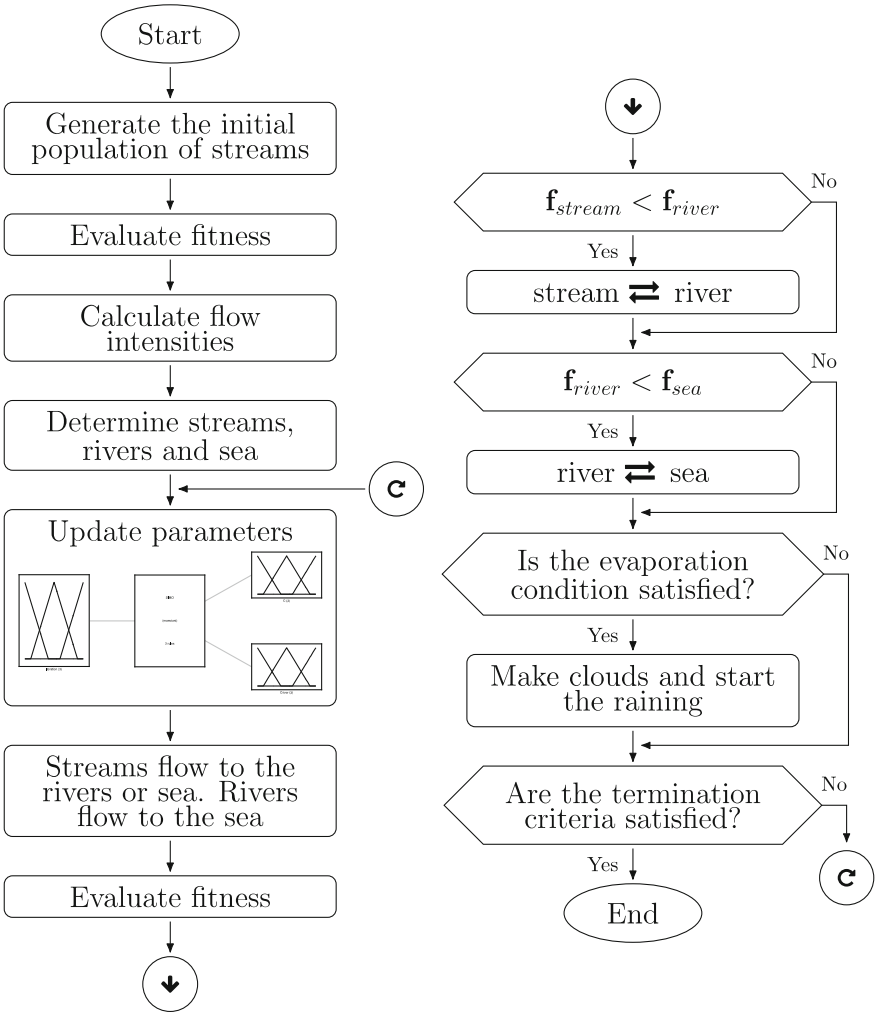

Fig. 6 Flowchart of the water cycle algorithm with fuzzy dynamic adaptation of parameters

Table 1 Test functions used for comparisons

No.	Function name	Equation	Domain
1	Michalewicz	$f(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot \left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}, \quad m = 10$	$0 \le x_i \le \pi$
2	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} [(x_i - 1)^2 + 100(x_{i+1} - x_i^2)^2]$	$-5 \le x_i \le 5$
3	De Jong	$f(x) = \sum_{i=1}^{n-1} x_i^2$	$-5.12 \le x_i \le 5.12$
4	Schwefel	$f(x) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$-500 \le x_i \le 500$
5	Ackley	$f(x) = -20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + \exp(1)$	$-32.768 \le x_i \le 32.768$
6	Rastrigin	$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$	$-5.12 \le x_i \le 5.12$
7	Easom	$f(x) = -\cos(x)\cos(y)\exp[-(x-\pi)^2 + (y-\pi)^2]$	$-100 \le x, y \le 100$
8	Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 \le x_i \le 600$
9	Shubert	$f(x) = \left[\sum_{i=1}^{5} i\cos(i + (i+1)x)\right] \cdot \left[\sum_{i=1}^{5} i\cos(i + (i+1)y)\right]$	$-10 \le x, y \le 10$
10	Yang	$f(x) = \left(\sum_{i=1}^{n} x_i\right)\exp\left[-\sum_{i=1}^{n}\sin(x_i^2)\right]$	$-2\pi \le x_i \le 2\pi$

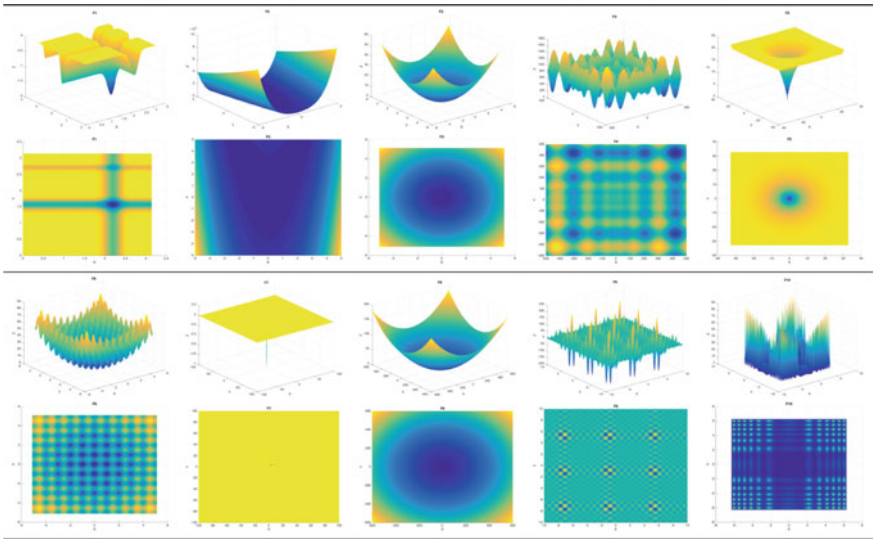

Fig. 7 Test functions plots for 2-dimensions, shown from a 3D perspective and from above

Table 2 MAE of 100 experiments for each test function in 2-dimensions

No.	Function	WCA	WCA-DPA
1	Michalewicz	6.28×10^{-14}	2.66×10^{-15}
2	Rosenbrock	1.36×10^{-9}	2.92×10^{-11}
3	De Jong	1.72×10^{-35}	3.18×10^{-119}
4	Schwefel	6.16×10^{-13}	4.90×10^{-13}
5	Ackley	3.43×10^{-8}	3.77×10^{-15}
6	Rastrigin	7.81×10^{-14}	1.42×10^{-16}
7	Easom	1.26×10^{-14}	5.55×10^{-18}
8	Griewank	8.88×10^{-4}	8.14×10^{-4}
9	Shubert	1.78×10^{-8}	1.92×10^{-44}
10	Yang	8.83×10^{-6}	8.83×10^{-6}

$N_{pop} = 50$, $N_{sr} = 7$, $d_{max} = 1 \times 10^{-6}$, $N_{it} = 4000$

Experiments in 2 and 30 dimensions were performed. The experiments consisted of 100 samples, taking the mean absolute error (MAE) as the measure of performance. Optimization results for the experiments in 2 and 30 dimensions are listed in Tables 2 and 3, respectively. Also, the parameters used are listed at the end of each table. From the results obtained in 2 dimensions (Table 2), we can observe improvements in most of the test functions. However, for 30 dimensions (Table 3), although it performed better for most of the test functions the results are only slightly better.

Table 3 MAE of 100 experiments for each test function in 30-dimensions

No.	Function	WCA	WCA-DPA
1	Michalewicz	1.13×10^{1}	7.38
2	Rosenbrock	4.53	7.88
3	De Jong	1.68×10^{-9}	3.24×10^{-8}
4	Schwefel	4.25×10^{3}	3.94×10^{3}
5	Ackley	4.13	2.12
6	Rastrigin	1.07×10^{2}	1.02×10^{2}
7	Easom	–	–
8	Griewank	2.75×10^{-2}	1.32×10^{-2}
9	Shubert	–	–
10	Yang	$7.82 \times \mathrm{lO}^{-13}$	1.14×10^{-13}

$N_{pop} = 50$, $N_{sr} = 7$, $d_{max} = 1 \times 10^{-6}$, $N_{it} = 4000$

6 Conclusions

From the experiments, it can be concluded that dynamically adapting the parameter C, can help to improve the performance of the WCA. But establishing an interval for the outputs, to work with any number of dimensions for the type of FIS developed could be a difficult or impossible task. It would be more appropriate not to bind the outputs to a given interval. Instead, could be better to have as outputs an increment or decrement of the parameter. Similar to the improved versions of particle swarm optimization: APSO or FAPSO developed in [13, 17], respectively. Another alternative could be to add a velocity component to the equations that update the positions of the streams, and then adapt an inertia weight instead of actual value C.

References

1. Atashpaz-Gargari, E., Lucas, C.: Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In: IEEE Congress on Evolutionary Computation. pp. 4661–4667 IEEE (2007).
2. Eskandar, H. et al.: Water Cycle Algorithm - A Novel Metaheuristic Optimization Method for Solving Constrained Engineering Optimization Problems. Comput. Struct. 110-111, 151–166 (2012).
3. Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science (80-.). 220, 4598, 671–680 (1983).
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1989).
5. Hosseini, H.S.: Problem solving by intelligent water drops. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. pp. 3226–3231 (2007).
6. Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer and its adaptive variant. IEEE Trans. Syst. Man, Cybern. Part B. 35, 6, 1272–1282 (2005).
7. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. pp. 1942–1948 (1995).

8. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Mach. Stud. 7 (1), 1–13 (1975).
9. Melin, P. et al.: Optimal Design of Fuzzy Classification Systems Using PSO with Dynamic Parameter Adaptation Through Fuzzy Logic. Expert Syst. Appl. 40, 8, 3196–3206 (2013).
10. Mirjalili, S. et al.: Grey Wolf Optimizer. Adv. Eng. Softw. 69, 46–61 (2014).
11. Sadollah, A. et al.: Water cycle algorithm for solving multi-objective optimization problems. Soft Comput. 19, 9, 2587–2603 (2015).
12. Sadollah, A. et al.: Water cycle algorithm with evaporation rate for solving constrained and unconstrained optimization problems. Appl. Soft Comput. 30, 58–71 (2015).
13. Shi, Y., Eberhart, R.C.: Fuzzy adaptive particle swarm optimization. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. pp. 101–106 vol. 1 (2001).
14. Shreve, R.L.: Infinite Topologically Random Channel Networks. J. Geol. 75, 2, 178–186 (1967).
15. Shreve, R.L.: Statistical Law of Stream Numbers. J. Geol. 74, 1, 17–37 (1966).
16. Yang, X.-S.: Flower Pollination Algorithm for Global Optimization, (2013).
17. Zhan, Z.H. et al.: Adaptive Particle Swarm Optimization. IEEE Trans. Syst. Man, Cybern. Part B. 39, 6, 1362–1381 (2009).
18. Zheng, Y.-J.: Water wave optimization: {A} new nature-inspired metaheuristic. Comput. {&} {OR}. 55, 1–11 (2015).